Az Országos Commodore Egyesület lapja

C újság

1992/5



Nyelvek világa

Programok

Még többet ésszel

BASIC-ben



EFAX-101

Műszaki paraméterek:

- Postai engedély
- Egyszerű kezelhetőség
- Egyszerű kivitel
- Automatikus és kézi vezérlés
- Másolási funkció
- Nagy átviteli sebesség (20 mp/oldal)
- Automatikus adássebesség (9600/7200/4800/2400 bps)
- Azonosító adatok közlése (fejléc, dátum, cím, telefonszám)
- Beépített óra
- Kontrasztállíthatóság
- Normál és finom felbontás
- G 3 CCITT kompatibilis
- 216 mm x 30 m faxpapír
- Programozási funkciók (1—10 csengetésszám, adás-vételi riport)
- Automatikus tesztprogram
- Automata FAX — üzenetrögzítő átkapcsoló



Avatex

Viszonteladóknak kedvezmény!

Telefonok, üzenetrögzítők, faxok, kis és nagy központok, kábelek, alkatrészek stb. — Tv-k, videók, kamerák, autórádiók!

SZERENCSEJÁTÉK RT

MIT, HOGYAN, HOL, MIKOR?

EGYESÜLETI ÜGYEK: Egyesületünknek tagja lehet mindenki, aki a tagsági díjat befizeti. A tagdíjat személyesen az egyesület irodájában (1025 Budapest, Vöröstorony utca 29. Telefon: 1-76-22-57), vagy átutalással az MNB 217-98 292, OTP 565-3610-8 számlára lehet befizetni. Megrendelés esetén számlát küldünk.

Pötyögőszolgálatunk valamint a szervizkedvezmény és az apróhirdetés lehetősége tagjaink rendelkezésére áll.

A DEÁKPÁHOLY tagjai minden hónapban megkapják a C-újságot, a tagsági díj egy egy évre 777 forint.

A PLUSZPÁHOLY tagjai minden hónapban megkapják a C-újságot, és kapnak havonta 3 db vásárlási utalványt. A tagsági díj egy évre 1888 Ft.

A SZUPERPÁHOLY tagjai havonta 15 példányt kapnak a C-újságból, és ezzel havonta 15x3 db vásárlási utalványt is. Az éves tagsági díj 20 900 Ft.

ÜGYFÉLFOGADÁS: Minden kedden és csütörtökön 12—16 óra között várjuk tagjainkat és az érdeklődőket.

PÖTYÖGŐSZOLGÁLAT: Az újságban megjelenő programokat másolja a megrendelők részére. Megrendelhető személyesen az egyesület irodájában vagy postai utánvéttel. Postacím: 1388 Budapest 62., Postafiók: 86.

APRÓHIRDETÉS: Az egyesületi tagoknak ingyen áll rendelkezésére. Nem tagoknak a hirdetés ára 100 forint. A hirdetés módja: az újságban megjelenő nyomtatvány kitöltésével.

A C-ÚJSÁG RÉGEBBI SZÁMAI megvásárolhatók az egyesület irodájában, vagy megrendelhetők utánvéttel.

Kedvezményes ár! Tagoknak olcsóbb!

Az újságban eddig megjelent programok gépenként összegyűjtve megrendelhetők. VC 20, C16, PLUS/4, C128, C64. További felvilágosítást is adunk a 1-76-22-57-es telefonszámon vagy levélben!

Vidéki pluszpáholy-tagjaink háromhavi tikett összegyűjtésekor igénybe vehetik a NOVOTRADE 2C Áruház csomagküldő szolgálatát.

VIDÉKEN TOVÁBBI INFORMÁCIÓK KAPHATÓK:
Baja, AXIS Kft.,
Győri Bartók Béla Művelődési Ház,
Jászberényi Városi Könyvtár,
Kecskemét, SZIGMA—BIT,
Pécsi Apáczai Csere János Gimnázium,
Zalaegerszegi Ságvári Endre Gimnázium.

Az Országos Commodore Egyesület módszertani kiadványa

Egyesületi iroda és szerkesztőség:
1025 Budapest, Vöröstorony utca 29. Tel./FAX: 1-76-22-57
Felelős kiadó: Horváth Judit, az egyesület elnöke
Főszerkesztő: Rados Péter, az OCE főtitkára
Felelős szerkesztő: dr. Horváth András
Művészeti szerkesztő: Bausz Sándor
Levélcím: Commodore Újság, 1388 Budapest, 62. Pf.: 86.
Index: ISSN 0237-756 X
Terjeszti a Magyar Posta
Megvásárolható a hírlapárusoknál
92.0027 MSZH Nyomda és Kiadó Kft., Budapest
Felelős vezető: Nagy László

Kedves Tagtársaink, Olvasóink!

Egyesületi és szerkesztőségi CÍMÜNK MEGVÁLTOZOTT!

1992. február 1-jétől új címünk:
1025 Vöröstorony utca 29.
Telefon: 1-76-22-57.
Megközelíthető a Batthyány térről a 11-es autóbusszal. A Kapy utcai megállónál kell leszállni. Elképzelhető, hogy a költözés miatt kicsit akadozni fog a kapcsolattartás Önökkel. Ezért megértésüket és türelmüket kérjük. Lapunk megjelenésében fennakadás nem lesz.
Postacímünk változatlan:
1388 Budapest 62., Postafiók: 86.

OCE

Pötyögőszolgálat

A kiválasztott programok megrendelhetők levélben, utánvéttel.
Postacím:
1388 Budapest 62., Postafiók: 86.
Felvilágosítás telefonon:
1-76-22-57.

Tisztelt Szerkesztőség!

Mellékelten megküldöm egy olyan programom listáját, aminek elkészítésére program másolók és szerkesztők talán gondoltak, de a megalkotása nem is olyan egyszerű! A lapban az utóbbi években ilyen feladatokat végrehajtó programot nem olvastam. Közlése örömet okozhat a HELP+ és SIOMON'S BASIC-al rendelkezőknek is.

Programom (ahogy van) a sorszámokat 100—300 között 10 lépésközzel készíti, de tetszőlegesen szabályozható!

A program képes kívánt BASIC sorszámok elkészítésére és törlésére annak alapján, hogy:
100: ⟨RETURN⟩ 100-as sor beírása;
100 ⟨RETURN⟩ 100-as sor törlése.

Az elkészítéshez meg kellett oldani az aktuális sorszám átmeneti tárolását a BASIC területén kívül, ahol nem zavarhatja az a BASIC programok, és felhasználni kívánt gépi rutinok működését. Szabad RAM terület található a 820—827 címeken. E címeket választottam a sok lehetőség közül, mert rövid, és valószínű mindenki kikerülheti ezek egyéb célú használatát.

A sorszámot két értékben (alsó és felső byte formában) kell tárolni, mert egy (POKE) címre beírható maximális érték 255.

Megoldása: tárolás előtt felbontjuk és képernyőre írás előtt újraalkotjuk a számokat.

A program működésének alapja, hogy a kívánt számot a képernyő meghatározott helyére írjuk, és egy trükkel a RETURN billentyű kétszeri lenyomását szimuláljuk (63992 sorban). Az END hatására végre is hajtódik a két RETURN, és beírja (törli) a memóriába(n) a képernyőre felírt sort; valamintfolytatja a feladatot a parancsként felírt GOTO63987-el az általunk meghatározott lépésközzel (63991 sorban) a kívánt maximális érték (63989 sorban) eléréséig. A meghatározott érték maximum 63950 lehet (63988 sor), hogy így ne zavarhassa e program működését.

Ha nincs szükség rá a továbbiakban a programot töröljük ki! Rövidsége miatt ez nem okozhat nehézséget.

Ám programszerkesztés közben ismételten is használhatjuk a megfelelő (kezdő/lépés/befejező) értékek beírásával, illetve törlés/írás választásával. Vigyázzunk, mert nem megfelelő értékek esetén a BASIC utasításaink helyett könnyen ″:″-t találunk! Nem a sor törlődik, hanem a program. Ennek kivédését szolgálja a 63980 sor. Szerkesztése esetén innen a REM-et vegyük ki, és az alapértékek beírása, ellenőrzése után GOTO63985-al indítsunk! Így nem lehet véletlen, csak tévedés! Természetesen ez lehetővé teszi programunk (egy része, szubrutinunk) törlését is a sorszámok egyidejű meghagyásával vagy nélküle.

A programsorok törléséhez egyszerűen vegyük ki a ″:″-t a 63990-es sorból (″ ″ maradhat), kezelése ugyanaz, mint az írásé.

A program előnye, hogy tetszőleges mennyiségű sor egy paranccsal törölhető és beírható, illetve programrész törölhető a sorok meghagyása mellett is. 1000 sorszámot kb. 5 perc alatt rögzít egyre lassuló tempóban, ahogy a memória telik, és nem hibáz!

Hátránya, hogy a meghatározott helyre a programba kell beírni a megfelelő adatokat. Ez azonban — a magyarázott lista alapján — gyakorlatlan programozónak sem okozhat nehézséget. Ha a megengedett érték helyére — használat után — 0-t írunk, semmi baj nem érheti programunkat!

Egy érdekességre is felhívom a figyelmet. A bonyolult program írása során (nem véletlenül) csak egy változót PS (programsor) használtam. Minden bonyodalom elkerülése érdekében (mivel GOTO-val is indíthatunk a változók értékei maradnak) PS használatát saját programban kerüljünk.

A program sokoldalúan használható! Ha soraink azonos BASIC utasítást használnak, a kettőspont 〈″:″〉 helyére írhatjuk azokat. A sorszámmal együtt az utasításoknak el kell férni egy sorban a képernyőn. Idézőjelen belül használhatjuk az utasítások rövidített formáját! Ellenőrizzük ezt az alábbiak beírásával:

63990 printps″pR1,″:print″goto63987″:rem nyomtatás
63990 printps″dA , , , , ,″:print″goto63987″:rem daták
63990 printps″?:?:?:?:?:?″:print″goto63987″:rem printek,

A végrehajtást a képernyőn ellenőrizhetjük; ha hibáztunk STOP billentyűvel állítsuk le és kezdjük újra a hibás sorok kitörlése után. Ha nem a sorszámban tévedtünk elég a javítás, mert átírja.

A magyarázat a programhoz viszonyítva talán hosszú, ám a szakszerű használathoz a működés megértése is fontos. Érdekes lehet sok olvasó számára a működése és kivitelezése egyaránt, mivel új programozási fogásokat ismerhetett meg.

Gépeljük be a programot, ügyeljünk a pontosságra! Ha a színkódokat kihagyjuk a 63990 sorban láthatóvá válik a másik felírt parancs (más alapszínű képernyőn is). RUN után elkészülnek a sorok. Kettőspont kitörlése és RUN után törlődnek a sorok! Saját igény szerint a REM-ek elhagyhatók.

Jó kísérletezést, eredményes munkát kívánok a lap olvasóinak!

Tisztelettel:

Mesterházi Sándor

```
63980 REM PRINT"TILOS TERULET":STOP
63981 REM *** C64 BASIC SORSZAMHELP ***
63982 REM KESZITETTE: MESTERHAZI SANDOR CELLDOMOLK
63983 REM JAVITSD AZ ALAPADATOKAT, UTANA RUN
63984 REM IRAS=PS"█ :" TORLES=PS"█" VAN 63990-BEN
63985 PRINTCHR$(147)
63986 PS=100:GOSUB63993:REM ELSO SOR(100)
63987 PRINT"█QQ":PS=PEEK(821)+256*PEEK(820):REM KIOLVASAS, SZAMITAS
63988 IFPS>63950THENPRINT:PRINT:END:REM MAX.SORSZAM
63989 IFPS>300THENPRINT:PRINT:END:  REM KERT MAX.SORSZAM(300)
63990 PRINTPS"█ :":PRINT"█GOTO63987█":REM KEPERNYORE IRAS/TORLES
63991 PS=PS+10:GOSUB63993:          REM LEPESKOZ(10)
63992 POKE631,13:POKE632,13:POKE198,2:PRINT"█":END
63993 POKE820,INT(PS/256):POKE821,PS-256*PEEK(820):RETURN:REM BONTAS,TAROLAS

READY.
```

Tisztelt Szerkesztőség!

Az év elejétől veszem rendszeresen a C-újságot és sok érdekes programot, ötletet találtam benne.

Azt a tényt, hogy most levélírásnak adtam magam, két tényező (az öröm és a sajnálkozás) együttes hatása okozta.

Az öröm amiatt, hogy cikksorozatban kezdték el a gépi kódú programozás ismertetését és a szerzők számítanak az olvasó közreműködésére, problémáinak megismerésére.

A sajnálkozás pedig azért, mert egyes programok esetében

— nyomtatási hiányosságok (1991/b. számban az UNNEW-64 esetében a data sorok elmosódottak) vagy
— a program futtatásával kapcsolatos értelmezési problémák (az 1991/6. számban az AUTOSCREENOFF programnál) merülnek fel.

Fenti száraz tényeket egy kissé bővebben kifejtve azt szeretném, ha a cikksorozat közérthető lenne olyanok számára is, akik értenek már valamit a számítógéphez, de még nem professzorai pl. a COMMODORE 64-nek. Ehhez csak arra van szükség, hogy a cikk írói ne szégyelljenek egyszerűen fogalmazni. Ha az így leírtakat zárójelben vagy lábjegyzetben kiegészítik a szakmai kifejezéseikkel, akkor segítenének igazán a szakmai dolgok megismerésében és nem szegnék kedvét az olvasónak.

Ebben nagyon reménykedem!

A másik témában jó lenne a nyomdát minőségi munkára kényszeríteni, mert ellenkező esetben marad a bosszúság, hogy a közölt programot az olvasó nem tudja használni (szerencsére az UNNEW-64-nél csak egy data érték hiányzott — de több elmosódott —, amit így ki lehetett számítani).

Szólni szeretnék még a programok futtatásával kapcsolatos kérdésekről és csak példaként említem az esetet, hasonló máskor, más helyen is előfordulhat.

Az AUTOSCREENOFF (26. old.) programnál értelmezési gondjaim vannak a leírással kapcsolatban. Az inkriminált mondat ,,... Adjuk be hát az AUTOSCREENOFF programot és mentsük ki lemezre...". Ezt én úgy értelmezem, hogy begépelem a programot, lemezre mentem későbbi használatra. De ha most elindítom a programot, a képernyő szépen kikapcsol, csak ha egy billentyűvel visszalépek, a látható képernyő a beállított idő után (legyen ez a legrövidebb egy perc) automatikusan ismétel és zavarja a folytatást. Pedig itt is olvasható a megjegyzés, hogy ,,... az adott gombnyomás ne legyen hatással az éppen futó programra...", amit nem tudok mire vélni. Aztán mi legyen a javasolt kezdő cím, amikor a kérdéses programé 2049, ami azonos sok program kezdő címével?

Két dologra tudok csak gondolni. Vagy rosszul értelmezem a sokszor túl tömören leírt magyarázó szöveget, vagy a magyarázatok egyszerűen félreérthetőek.

Egyébként a ,,Gépi kódú programozás"ban, az időmérő sorban is van nyomdai hiba, ami értelemszerűen ,,... clr ..."-ra javítandó és csak érdekességként jegyzem meg, hogy C64-nél a futási idők 325, 26, 13 másodperc.

Befejezésül arra kérem Önöket, ha lehet, válaszoljanak levelemre.

Üdvözlettel:

Cakó János

Válasz Cakó Jánosnak

Az ,,AUTOSCREENOFF" program leírásánál jól értelmezte, valóban ki kell menteni a beírt programot, ahogy Ön írta: ,,későbbi használatra". Ez egyébként minden programra igaz, hiszen, ha futtatjuk a programot bármi történhet: például valamit elhibáztunk a beírásnál és ,,eltűnik" a program a memóriából. Kezdhetjük elölről a beírást. Ennél ugye jobb, ha visszaolvassuk ilyenkor az elmentett programot, megkeresssük a hibát, kijavítjuk, újra elmentjük, újra futtatunk.

A program működésével kapcsolatos kérdésére következő a válaszom. Ellenőriztem a programot és nálam tökéletesen működött. Feltételezem, hogy Önnél a beírásnál hiba történt. Ezért javaslom, hogy egészítse ki a programot a következő részekkel.

Szúrja be a " READY Y:" után " C=C+Y:" értékadást.

Írjon be egy új sort:

255 IF C ◊ 7994 THEN PRINT " HIBA A DATA-SOROKBAN..." : END

Ezzel ellenőrzi, hogy valóban helyesek-e a DATA-sorok adatai.

Bízom benne, hogy ezek után jól fog működni a program.

Kovács Mihály

Tisztelt Szerkesztőség!

Ezzel a rövid programmal érdekes betöltőt készíthetünk programjainknak. Ha más szöveget akarunk vele kiíratni akkor a 10-es sorban a számokat kell csak felcserélnünk a következő módon: a = 1, b = 2, space = 32. Egyszerre 21 db betűt tudunk kiíratni.

Íme a lista:

Mahunka Zsolt

```
0 POKE 53281,0:POKE53280,2
10 DATA6,15,16,18,15,7,18,1,13,32,9,19,32,12,15,1,4,9,14,7,-1
11 PRINT"<CLR>"
20 FOR X=1714 TO 1714+21
25 POKEX,62 :IF B=7 THEN POKEX,32
27 FOR P=1 TO 100:NEXT
30 READ B:IF B=-1 THEN 1000
35 POKE X,B+128
40 NEXT
50 END
1000 LOAD"FOPROGRAM",8

READY.
```

Tisztelt Szerkesztőség!

Köszönöm, hogy válaszoltak levelemre, és küldték az újságokat, hogy segítsenek.

Mellékelten küldöm a matematikai programomat lemezen. Ez nyomtatásban nem hiszem, hogy megjelenhet, hisz sok a felhasznált grafika, hogy a külalak tetszetős legyen. Így csak lemezen lehet lefuttatni, és ha valakit érdekel, lemez ellenében szívesen rendelkezésre bocsájtjuk. Ugyanis nem egyedül készítettem, Erdélyi Gyula villanyszerelő ismerősöm volt segítségemre, hogy a tetszetős formát megvalósíthassam. Én a matematikai magot és a programot adtam, így kettőnk munkája.

A kezelés nagyon egyszerű, olyan gyerek is tudja kezelni, aki először ül géphez. A betöltés egy kissé hosszadalmas, de utána ez a munka megtérül.

A program leírását és kezelési útmutatóját mellékelem a záródolgozatomban írt ezen részével és a tapasztalataimmal

Én a számítógépet az óráimon is szeretném használni s ezért lenne szükségem C64-re írt alsó tagozatos oktató programokra. Remélem tervemet sikerül megvalósítani és az oktató-nevelő munkában e téren is érhetek el eredményeket.

Szívességüket mégegyszer köszönöm!

Tisztelettel:

Cziklár Károlyné
tanító

A program leírása

A betöltés után megjelenik a fő cím: Matematikai programok gyűjteménye. A cím alatt van a két program neve. Főmenüje: alapműveletek gyakorlása 1. Római—arab számok. 2. Alatta található az utasítás: Nyomd le a megfelelő billentyűt! Az 1-es szám leütése után jelenik meg az alcím: Matematikai alapműveletek gyakorlása és vizsgázás. Bármely billentyű leütése után látható az almenü: összeadás, kivonás, szorzás, osztás, 4 alapművelet (kicsit lejjebb), gyakorlás, vizsga. Az almenüben a 4 alapműveletet lehet külön-külön, valamint vegyesen gyakorolni, ill. ellenőrizni lehet, milyen szintre jutott a tanuló a számolási gyakorlatban.

Az első 5 lehetőség közül úgy választhat a tanuló, hogy a CRSR gombbal a kívánt részre állítja a fényt. A gyakorlás, vagy a vizsga kiválasztásához a szóköz billentyűt kell használni. A kívánt részek kiválasztása után a Return billentyű leütése után következnek a feladatok. A műveleteket az ezres számkörben végzik a tanulók. Az összeadás gyakorlásánál megjelenik a keretben a két összeadandó és az egyenlőség jele. A tanuló a kiszámított összeget a számbillentyűkkel beüti és ha helyes a megoldás, megjelenik a kiírás: Jó, ügyes vagy! Akarsz még gyakorolni? (I N). Ha tovább szeretné gyakorolni az összeadást, az I betű leütése után ismét új összeadandók jelennek meg. (A számok véletlenszerűen jelennek meg.) Ha nem jó a megoldás, a következő kiírás jelenik meg: Nem jó, próbáld újra! és ismét ugyanaz a két szám jelenik meg. Ha harmadszori próbálkozásra sem jó a megoldás, ismét kiírja: Nem jó! A jó eredmény: és a tanuló láthatja a helyes megoldást.

Az N billentyű leütése után ismét az almenü jelenik meg és választhat a tanuló.

A többi műveletnél is ugyanez az eljárás. Ha együtt szeretné a 4 alapműveletet gyakorolni, akkor az 5. megoldást választja a tanuló, vagy a nevelő.

A vizsgánál is ugyanez a sorrend, csak abban különbözik, hogy itt a tanuló már az első helytelen számítás után új feladatot kap, tehát nincs javítási lehetőség. 10 feladatot kell megoldani, minden jó megoldás 1 pontot ér. Ha befejezte, a 10. művelet után láthatja a képernyőn, hogy hány pontot szerzett és ez milyen osztályzatot jelent.

A vizsga részt a 4. osztályban év eleji felmérésként is használhatja a nevelő, hisz rövid idő alatt lemérheti, hogy tanulói az 1000-es számkörben milyen szinten vannak a fejszámolás terén.

Római számjegyek írása, olvasása

Filep László—Bereznai Gyula: A számírás története c. könyvében kifejti, hogy a római számok elnevezése nem helyes, bár elterjedtsége miatt gyakran használjuk. ,,A számok ugyanis nem rómaiak, arabok vagy egyiptomiak, nincs nemnzetiségük, egyszerűen csak számok, írásmódjuk lehet római, arab, egyiptomi stb. A római számok helyett tehát jobb a római számírás, római számjegyek elnevezéseket használni."

Ennek az ismeretnek a birtokában kezdtem el foglalkozni a tanításban előforduló római számírás követelményeivel, ami csak tananyagként szerepel a tantervben, nem követelmény egyik osztályban sem. Ennek ellenére a római számírás az emberi kultúra olyan lényeges rekvizítuma, amelynek mellőzése vétek volna tanítványaink műveltsége ellen. Sőt a mindennapi gyakorlatban is megtalálhatók a római számok. Pl.: a hónapok jelölésére, Budapest kerületeinek megkülönböztetésére használatosak, órák számlapjai is gyakran készülnek római számokkal, műemlékpületek felíratai is tartalmaznak római számokat. Az idő mérése viszont követelmény.

A római számírás nem a helyiérték-rendszer alapján történik, ezért a római számokkal a műveletek elvégzése nagyon bonyolult. Ez a magyarázata annak, hogy a római számok már csak mint sorszámnevek játszanak szerepet. Viszont a helyiérték-rendszer előnyeinek szemléltetésére jól használhatók. Motivációs célra is alkalmas a római számokkal való foglalkozás. Nagyon szeretik a gyerekek a római számokkal megadott fejtörőket, amelyek pl. 1—1 pálcika (gyufaszál) átrakását tűzi ki feladatul.

A római számírás jegyei közül az ötszázas és az ezres kerül bevezetésre a 3. osztályban. 1000—2000 közötti esetekben a 4. osztályban gyakoroljuk.

Célszerű megvizsgálni a tízeseknek és a százasoknak az egyeskével anlóg képzését:

I	II	III	IV	V	VI	VII	VIII	IX
X	XX	XXX	XL	L	LX	LXX	LXXX	XC
C	CC	CCC	CD	D	DC	DCC	DCCC	CM

Vannak tanulók, akik hamar megértik a római számképzést és szívesen gyakorolják ezt. Ezeknek a gyerekeknek a kedvéért készítettük el azt az oktatói programot, ami ismereteket is nyújt a gyerekeknek, de gyakorolhatnak is, sőt még vizsgázhatnak is a római szám ismereteiből.

A matematikai programok gyűjteményében a 2-es szám mellett található a Főmenü elnevezés: római—arab számok. A 2-es szám leütése után megjelenik a cím: Római—arab számok gyakorlása, majd utána az almenü: Római—arab számolás, római gyakorlás, arab gyakorlás, római—arab gyakorlás. A menü első részét kiválasztjuk a CRSR gomb segítségével és a Return leütése után látjuk a feladatot. Ez a gyakorlás, ahol tetszés szerint lehet római vagy arab számot beütni és megkapjuk a megfelelő-

jét. Ebből a részből a balra mutató nyíl billentyűjével lehet kilépni és ismét az almenüből választhatunk. A munka menete megegyezik az alapműveleteknél alkalmazott eljárással. A vizsgára úgy térhetünk át, ha a gyakorlás bármelyikére állítjuk a cursort és lenyomjuk a szóköz billentyűt. Ezután átvált vizsgáztatásra, ami szintén ugyanaz, mint az alapműveleteknél.

A számolási (gyakorlási) rész alkalmas arra, hogy a tanuló, ha nem ismeri a római számokat, vagy elfelejtette azt, a géptől kérhet segítséget. Ezt akkor is megteheti, ha a gyakorlás részben dolgozik. Ehhez természetesen az szükséges, hogy ezzel a programmal többször találkozzon a tanuló és tudja annak egyszerű, de több gyakorlást igénylő kezelését.

Azért is célszerű ezzel többször is foglalkozni, hisz a számírásban Magyarországon a XV. századig a római számjeleket használták. Az új számírás legkorábbi emléke 1407-ből való. A Mátyás templomon látható Országcímeren az évszám már arab számokkal van felírva (1470).

Tapasztalatok, felhasználási javaslatok

A matematikatanítás folyamatában gondoskodnunk kell a tanulók bizonyos jártasságainak és készségeinek kialakításáról. Ezeket a tantervek osztályonként és témakörönként pontosan előírják.

,,Jártasságon a tanulók matematikai tevékenységének azt a fokát értjük, amelyben a tanuló új feladatok megoldására képes, meglevő ismereteinek alkotó, kombinatív felhasználása révén."

,,Készségen a tanulók tudatos matematikai tevékenységének automatizált komponensét értjük, amely a tevékenység többszöri ismétlése, gyakorlása során alakul ki." (Nagy Sándor: Didaktika)

A 3. osztályos tanulónak az ezres számkörben a szóbeli összeadás, kivonás, szorzás és osztás készségével kell rendelkeznie. Hogy ezt megvalósítsuk, úgy kell az általunk megismert és jónak felismert módszerek között válogatnunk, hogy azok a legeredményesebbek legyenek ezen a területen. A módszerek mellett jelentős szerepet kap a tanulás eszköze is. A számítógép elterjedése bővítette az eszközök skáláját, amit akkor érdemes használni, ha a felhasználó programok a munka eredményességét tükrözik. Egy év távlatából ebből túl sokat lemérni nem lehet, csak egy-egy jelentősebb területet lehet kiemelni. Egyik ilyen része a motiváció. A jól megszerkesztett program, az azonnali visszajelzés, a közvetlen kapcsolat a tanulóval, mind sikeresebbé teszi az órát. Az öröm érzése tölti el a nevelőt, amikor látja, hogy a tanulók élvezik a gyakorlásnak ezt a módját és ösztönzi őket arra, hogy jobb és jobb eredményt érhessenek el.

Ebben a tanévben nem tanítok harmadik osztályban, így az előző részben ismertetett 2 programot órán kívüli foglalkozáson próbáltam ki, 5 harmadik osztályos tanulóval. A gyerekeket úgy választottam ki, hogy képességük különböző legyen. 2 tanuló még nem ült számítógépnél, de jó matematikából, 3 tanuló tudja a gépet kezelni, de különböző képességű.

Először a fejszámolást végző programot töltöttem be. Röviden elmondtam, hogyan kell a gépet kezelni, milyen billentyűket használjanak. Kértem, hogy először a gyakorlást végezzék el minden témakörben 5—5 feladatot oldjanak meg. Ezután hagytam őket önállóan dolgozni.

Azok a tanulók, akik ismerték a gépet csak 3 perccel előbb értek a feladatok végére, mint a másik két tanuló. 15 percet vett igénybe ez a feladat.

Ezután került sor a vizsgára, itt csak a 4 alapművelet vizsgáját kértem. Nagyon rövid idő alatt elkészültek, mindössze 6 percre volt szükség a 10 feladathoz és 4 tanuló 9—9 pontot ért el, 1 volt aki 8 pontot. Amikor a végére értünk, további ,,munkalehetőséget" ajánlottam fel, de természetesen a számítógép vonzereje győzött. A tanulóknak nagyon tetszett a program és igényelték, hogy matematikai órákon is hasonló feladatokat oldhassanak meg. Annyira lekötötte őket ez a munka, hogy a mellettük játékprogrammal játszó szakkörösök sem vonhatták el figyelmüket. Különösen a vizsgázás keltette fel érdeklődésüket. Itt tűnt ki, hogy érdekli őket a feladat, intellektuális örömet jelent számukra annak megoldása, szívesen foglalkoznak vele. Biztos benne a gyerek, hogy meg tudja oldani azt, vagy legalábbis reméli, és így előre élvezi a sikert, ami a feladat megoldásával jár. A munkakedv, a munka szeretete, önbizalom, igényszint, a teljesítmény emelésére való törekvés kialakítása nagy mértékben függ attól, hogy milyen mértékben teremtjük meg a szabad légkört, ami alkalmas arra, hogy a tanulók önálló kezdeményező magatartása kialakuljon.

Jelentősen emelhetjük a tantárggyal való foglalkozásra fordított energia mennyiségét, ha közvetlen, nem túlságosan távoli, jól körülhatárolt célokat tűzünk ki a tanulók elé. Itt pl. 1—1 alapművelet gyakorlása után a következő órán végezhetjük a vizsgáztatást. Ha a tanulót nem váratlanul fogja érni, hanem előre tudja, hogy a gyakorlást vizsga követi, amit értékelünk is, a tanulók igyekeznek minél több feladatot megoldani, sőt a biztos programkezeléssel előre, önállóan is képesek lemérni a vizsga segítségével tudásukat. Ez a tanulók nagy részét aktivizálni fogja.

A római számírásnál még nem tanulták a D-t és az M-et, sőt ebben a tanévben még nem is foglalkoztak vele. Azt hittem ez a program nem is fogja őket érdekelni. Tévedtem, mert ugyanolyan lelkesen dolgoztak vele, csak itt a számírás került előtérbe, ahol a gép adta a megoldást. Itt aztán próbálkoztak a nevezetes évszámokkal, a tárgyévvel, a születési évekkel, történelmi évszámokkal. Hamar rájöttek azonban a nagy számok képzésének a logikájára, így a gyakorlást és a vizsgát is élvezettel végezték később.

Itt látom én jelentőségét annak, hogy az órákon is ugyanilyen lelkesedésre lehetne a tanulókat rábírni, ha használnánk hasonló programokat.

A közeljövőben szeretnénk alsó tagozatos C64-re írt oktatóprogramokat szerezni és a kollégákkal is megismertetve ezek használatát, iskolánkban rendszeressé tenni az ilyen oktatást is.

A programírás nekem nagyon sok energiámba és időmbe telik. Ezt a két programot segítséggel készítettem el. Városunkban megismertem egy fiatal villanyszerelőt, Erdélyi Gyulát, aki hobbyból végez programozást. Ezt a fiatalembert vontam be ebbe a hasznosnak ígérkező munkába. Én a matematikai alapot adtam, Ő azt öntötte formába. A továbbiakban is szeretnénk együtt dolgozni és az alsó tagozatos tanulóinknak több olyan programot készíteni, ami hasonló érdeklődést, lelkesedést vált ki a tanulókból és remélem a tanító kollégáimból is. Szeretném, ha az eredményessége is pozitív lenne!

A számítógép (bár rövid az az idő, ami a tapasztalatot adja), az oktatás kiváló segédeszköze lehet. A tanítók szempontjából előnye, hogy erősen motiválja a tanulókat, és inkább le kell beszélni őket a géppel való munkáról, mint rá. A számítógép türelmes, és nem ad alkalmat személyes konfliktusokra, vagyis olyan, mint az ideális tanár. Az egyéni ütemű feldolgozást lehetővé tevő programok segítségével valószínűleg olyan tempó alakítható ki, amely a hatékony tanulás előfeltétele.

A számítógép azonnali visszacsatolást is biztosít, megerősítve azokat a fontos készségeket, amelyeket a tanulókban ki akarunk alakítani. Ebben elsősorban a gyakorló programok segíthetnek. Egyéni és csoportos munkára is alkalmasak. Különösen alkalmasak a számítógépek arra, hogy az osztály tehetséges ta-

nulóit lekössék, amíg a többieknek a tanító segítségére van szükségük. A nevelő számára nyilvánvalóak az iskolai számítógép-használat előnyei, de nem lehet különálló oktatási eszközként használni, hanem be kell építeni az elemi szintű oktatás elméleti rendszerébe. A hatékony számítógépes oktatás a szoftver minőségén áll, vagy bukik. A végső felelősség e téren is a pedagógusé, akinek a program megtervezőjeként össze kell fogni a számítógépes szakemberekkel, hogy a céljainak legmegfelelőbb program elkészülhessen.

A mai gyerekek lesznek a jövő számítógépes társadalmának tartópillérei. Hogyan érhetjük el, hogy sokkal korábban megbarátkozzanak a mikroszámítógépekkel, mint a szüleik? A számítógépeknek illeszkedniük kell az általános iskolai tananyaghoz.

A gyerekek körében nagyon népszerű és jól alkalmazható pl. a Logo programnyelv. Különösebb előtanulmányok nélkül, azonnali sikereket érhetnek el a képernyőre rajzolással. Papírral és ceruzával nem lehet elsajátítani ezt a nyelvet, feltétlenül „gépközelbe" kell lenni.

Az általános iskolai számítógépes oktatás bevezetése mellett szól az, hogy a gyerekekben kialakul az a szemlélet, hogy a számítógép ugyanolyan oktatási segédeszköz, mint pl. a könyv, amelyet többféle célra lehet felhasználni.

Az iskolába bekerülő számítógép a tanárra is feladatot ró. A gép hatékony alkalmazása teljesen annak a függvénye, menynyire ismeri a pedagógus a gép lehetőségeit. Nem feltétlenül szükséges mélyreható elektronikai vagy programozási ismeret! A tanár feladata, hogy a forgalmazott programokat hasznosítsa, olyan módszereket dolgozzon ki, amelynek alkalmazásával növelhető az oktatás hatékonysága, színvonala.

Tisztelt Szerkesztőség!

Gyakran foglalkozik az újság directory ötletekkel. Most egy olyan rutint küldök, amit a COMMODORE újság 1991/12. sz. 8. lap „Directory 3" alapján írtam. A lemezekről csak a „SEQ" filék neveit olvassa be, és azokat két oszlopban kiírja a képernyőre.

Néhány apróságot jegyeznék csak meg:

190-es sorban hurkot alkalmaztam az eredeti (180. sor) helyett, ezzel a beolvasás ideje majdnem felére csökkent;

200-as sorban válogatás történik;

220—230 sor biztosítja a pontos kiírást 99 blokkig (100-on felül új sort kell kreálni a 220. sor alapján);

270—290 sor írja ki az eredményt és törli is a kiírt változókat.

A küldött rutint adatkezelő programomban más apróságokkal együtt sikeresen beépítettem, könnyítve ezzel a felhasználók dolgát.

Természetesen más file-típusok olvasására is átalakíthatjuk.

Tisztelettel:

Mesterházi Sándor

```
10 REM SEQ ALLOMANYOK OLVASASA DIREKTORYBOL
20 REM KESZITETTE: MESTERHAZI S. CELLDOMOLK
100 DIMN$(145)
110 GOSUB160:REM RUTINHIVAS
120 END
130 :
140 REM LEMEZTARTALOM OLVASAS
150 :
160 PRINT"□A LEMEZEN LEVO SEQ ALLOMANYOK NEVEI":PRINT
170 OPEN15,8,15,"I":GOSUB340:REM HIBA
180 Z=.:OPEN1,8,.,"$":POKE781,1:SYS65478:GETA$,A$,A$,A$
190 FORK=.TO27:GETA$:N$(Z)=N$(Z)+A$:NEXTK
200 FORN=20TO23:IFMID$(N$(Z),N,3)="SEQ"THEN220
210 NEXTN:N$(Z)="":GOTO250
220 IFMID$(N$(Z),3,1)=CHR$(34)THENN$(Z)=MID$(N$(Z),3,18):GOTO240
230 N$(Z)=MID$(N$(Z),4,18)
240 Z=Z+1
250 GETA$,A$,A$,A$:IFST=64THENSYS65484:GOSUB340:CLOSE1:CLOSE15:GOTO270
260 GOTO190
270 FORK=.TOZ-1STEP2
280 PRINTN$(K);:PRINTTAB(20)N$(K+1):N$(K)="":N$(K+1)=""
290 NEXTK
300 PRINT:PRINT"BILLENTYURE VAROK"
310 GETX$:IFX$=""THEN310
320 RETURN
330 :
340 INPUT#15,AH,BH$,CH,DH
350 IFAH=.THENRETURN
360 PRINT"□LEMEZHIBA!"
370 SYSW,.,22,AH","BH$","CH","DH
380 GETX$:IFX$=""THEN380
390 CLOSE1:CLOSE15:END
READY.
```

Számítógépes fogalmak

Lexikon

IV. rész

Monitor

A monitor egy számítógépes rendszer képernyős eszköze. Elméletileg egy monitor megfelel egy televíziónak, viszont jobb minőségű képet ad és persze nincsen benne vevőrész (tuner) a televíziós adásokhoz. Különbséget kell tenni a monokróm (egyszínű) és a színes monitorok között. Egy monitor vételénél mindenképpen ügyelni kell arra, hogy az ,,megértse" azokat a jeleket, amelyeket a számítógép felé küld. Tehát a monitornak rendelkeznie kell egy megfelelő bemenettel. A hordozható számítógépek esetében a hely és a súly problematikája miatt más technológiát használnak (LCD képernyők stb.) a képábrázoláshoz. A nyelvhasználatban monitornak hívnak némely gépi kodú segédprogramot is (amelyekkel valahová ,,betekintést" szeretnénk nyerni), de ennek semmi köze sincs az általunk most tárgyalt monitorhoz.

Monokróm monitor

Mindazokat a számítógépes rendszerhez való képernyős készülékeket, amelyek csupán egy szín, és annak árnyalatai visszaadására képesek, monokróm monitornak nevezzük. A műszaki problémák egy ilyen készülék gyártásánál jóval kisebbek, mint a színes monitorok esetében. Többek között a monokróm képcsövek is sokkal olcsóbbak mint a színes készülékhez valók. Másrészről csupán a számítógépből érkező fényerő jelet kell kiértékelni. Éppen ezért a monokróm monitorok jóval olcsóbbak, mint a színesek, ha egy adott minőségi szintet nézünk. Általában a zöld szín dominál a monokróm monitorok esetében. Gyakori még a borostyánsárga, de vannak fekete-fehér monitorok is. Ezeket eleddig csupán a profi szövegfeldolgozó programoknál találhattuk meg, csupán az utóbbi időben jelentek meg sorozatban is olyan gépeknél, mint a Macintosh vagy az Atari ST.

Sávszélesség

A sávszélesség egy jel azon jellemzője, amelyet az átviteli frekvenciák legmagasabb és legalacsonyabb értékének különbségéből kapunk. Egy monitor sávszélessége lehetőleg olyan nagy legyen, amilyen csak lehet. 18 MHz-es sávszélesség esetében már jó 80 jel/soros ábrázolásra van mód. A tv-nél a sávszélesség általában 5—7 MHz, a nagyfelbontású (High Resolution) monitoroknál ez az érték minimum 20 MHz. Egy színes monitor ezzel a sávszélességgel azonban igen sokba kerül, ezért a házi computerek esetében ezek számításba se jöhetnek.

A sávszélesség az egyik legfontosabb kritérium a helyes monitor kiválasztásánál, de fontos az úgynevezett pixel (képpont), a színes monitoroknál a tripel távolság, valamint a maszk fajtája a képcsőben is.

Lyukacsos maszk, réselt maszk

Minden egyes színes képcső rendelkezik szorosan a frontlap mögött egyfajta szűrővel. Ezzel gátolják meg azt, hogy az elektronsugár nyaláb eltalálja a szomszédos sugárzott részt. Egy ilyen maszk nélkül a színes kép teljesen életlen és elmosódott lenne. Különbséget kell tennünk a lyukacsos maszk és a réselt maszk között. A lyukacsos maszkkal szemben a réselt maszk nem kerek, hanem hosszúkás nyílásokkal rendelkezik. Emiatt az írások visszadásánál kevesebb a törés által okozott torzítás, mivel a merőleges írási élek párhuzamosan tudnak kilépni. Manapság szinte csak réselt maszkokat használnak, amely jótékonyan hat a képminőségre. A nagyobb előállítási költségek miatt az ilyen monitorok azonban drágábbak mint az összehasonlítható lyukacsos maszkkal rendelkező modellek.

RGB

A számítógép által a rendelkezésre bocsátott RGB-jel egy megfelelő bemenettel rendelkező színes monitor számára információt ad a három primer szín — a vörös, a zöld és a kék — intenzitásáról. Az RGB (Red, Green, Blue) jelet elméletileg minden színes készülékhez használhatjuk, amelyik RGB vagy SCART bemenettel rendelkezik. Különbséget kell tennünk az analóg és a digitális RGB-jel között. Míg a digitális jelnél a színösszetevőket digitális impulzusok segítségével (low vagy high TTL jelszínt) ábrázoljuk, az analóg RGB-jel ugyanezt fokozatmentesen állítható jelszinttel teszi. Az előny: Mialatt a digitális RGB-jellel maximum 512 színt lehet ábrázolni, az analóggal elméletileg tetszőleges számút. A színek számát csupán a számítógép rendelkezésre álló memóriája korlátozza.

FBAS-jel

A BAS (Bildinhalt-Austast-Synchron, azaz képernyőtartalom letapogató szinkron) egy olyan fekete-fehér, illetve világos/sötét jel, amelyeket a fekete-fehér és a színes készülékek video része is ,,megért". Ha még a színinformációt is hozzávesszük, akkor beszélünk FBAS jelről. Az FBAS tartalmazza a szín és a fényesség információit, valamint a kép fölépítéséhez szükséges szinkronizációs jeleket is. A számítógépnek a FBAS jel előállításához először össze kell keverni a színjeleket, a monitornak pedig azokat egy dekóder kapcsolás segítségével ismét szét kell bontania. Ilyenkor persze az egyes összetevők minőségében romlás léphet föl, amely az életlenségben és a színek keveredésében mutatkozik meg. Ha nagyobb képfelbontásra van szükség, akkor az RGB jelet kell előnyben részesíteni.

Felbontás

A felbontás (Resolution), vagy más szóval a képfelbontás alatt azokat a pontszámokat értjük, amelyeket egy adott felületen egymástól optikailag még meg lehet különböztetni.

A legnagyobb felbontású grafikus számítógépek 4096×4096 pontot képesek ábrázolni, a házi számítógépek az átlagosnak mondható 320×160 képpont (pixelt) tudnak megjeleníteni. A pixel egy angol műszó, amely a „picture element"-ből lett képezve, ami magyarul képpontnak fordítható. Egy képernyő is csupán korlátozott felbontóképességgel rendelkezik, ami a sávszélességtől és a tripel távolságtól függ.

Egy színes monitor vételekor a tripel távolság a legfontosabb kiválasztási kritérium, ennek az értéknek lehetőleg kisebbnek kell lenni mint 0,4 mm. A tripel az a legkisebb színegység a képernyőn, ami egy vörös, egy zöld és egy kék színpontból áll.

PAL

A PAL (Phase-Alternation-Line), egyike a jelenlegi legbonyolultabb, de a legjobb színátviteli rendszereknek. Egy trükk segítségével az átviteli úton fellépő színhibákat ki lehet küszöbölni. Ehhez minden egyes sor után a színinformáció jel fázisát 180 fokkal elforgatjuk. Ha két egymást követő sor ugyanazzal a színhibával rendelkezik, akkor ezek így egymást közömbösítik. A PAL egy német fejlesztés, és az túlnyomórészt Európában terjedt el. A legrégibb, ugyanakkor a műszakilag legegyszerűbb rendszer az amerikai NTSC (National Television System Commitee), amelyet tréfásan Never The Same Colour-nak (sosem ugyanaz a szín) is szoktak nevezni, mivel ennél a rendszernél gyakorta lép föl az úgynevezett vörös/zöld ugrás. Franciaországban és a legtöbb kelet-európai országban a SECAM-ot használják, amely francia találmány.

Változók hozzá-rendelése másképpen

Még a legeslegkezdőbb C64-es is tudja azt, mik azok a változók és hogyan lehet nekik értéket adni. A legegyszerűbb azt közvetlenül a programban megtenni (konstans). Ezt az egyenlőségjellel ⟨=⟩ végezhetjük el:

```
A = 10:B$="HELLO" stb.
```

Ezenkívül biztos ismerjük már az INPUT és a GET utasításokat is. Ezek igénybe vételével a billentyűzeten keresztül tudunk értékeket hozzárendelni. A legtöbb esetben ez a leggyakorlatiasabb módszer. Ha azonban egy olyan programot írunk, amely igen sok fix értékkel (konstans) dolgozik, úgy a billentyűzeten keresztül történő bevitelnek nem sok értelme van, hiszen akkor a program minden egyes indításánál be kell adni a szükséges adatokat. Mindez pedig igen nehézkes dolog. Akkor inkább a "=" jeles megoldás kínálkozik. Ezzel elérhetjük, hogy a programunk elején a szükséges változók megkapják a kívánt értékeket, ahogy azt fentebb láthattuk. Ha például az A értékének 256-nak, az X értékének —45-nek kell lenni, akkor a programunk első sora így nézhet ki:

```
10 A = 256:X= —45
```

Ez a módszer mindaddig kellemes, amíg kevés számú ilyen értékadásra van szükség. Mihelyst azonban nagyobb adatmennyiséget kell feldolgozni, főleg ha például tömbváltozókról van szó, bizony megint csak a frász fog minket kerülgetni a rengeteg kényszerű gépelési munka miatt. Azért mondjuk, hogy ez a munka fölösleges, mivel a C64-es BASIC-je egy sor olyan parancsot bocsát a rendelkezésünkre, amellyel értéklistákat hozhatunk létre a programban, s amelyeket azután libasorban szépen hozzárendelhetünk a szükséges változókhoz. Ezeknek az adatlistáknak a működését egy kis példával szeretnénk szemléltetni:

Az A, a B és a C változókhoz az 5, 10, 15 értékeket akarjuk hozzárendelni. Az ismert módszer szerint a programunk így nézhet ki:

```
10 A = 5
20 B = 10
30 C = 15
```

Ha most beadjuk a RUN parancsot, minden változó megkapja a neki szánt értéket. Ezt úgy is ellenőrizhetjük, hogy beadjuk a PRINT A, B, C utasítást, mire a három változóhoz rendelt érték megjelenik a képernyőn.

Nincs azonban semmi akadálya annak, hogy a három értékből egy listát hozzunk létre. Töröljük hát az előbbi programocskát NEW-val, és adjuk be a következő sort:

```
10 DATA 5, 10, 15
```

Mihelyst a számítógép egy DATA parancsot talál, azonnal tudja, hogy itt egy értéklistáról van szó, ezért a következő BASIC sorra ugrik. Ezt azért fontos tudni, mivel amennyiben a DATA sor mögött egyéb BASIC parancsok találhatók, az interpreter azokat nem dolgozza föl.

Most pedig kiolvastathatjuk a géppel a fölállított listát, és az értékeket a kívánt változókhoz rendelhetjük. Ezt a READ paranccsal valósíthatjuk meg. Olvastassuk be tehát előbb a legelső értéket ⟨5⟩ az adott paranccsal az A változóba:

```
20 READ A
```

Ennek az BASIC sornak a végrehajtása után az A értéke 5 lesz, amelyet a PRINT A paranccsal ellenőrizhetünk is. A B-nek most a 10 értéket kell megkapnia. Ez is hasonlóan történhet:

```
30 READ B
```

Adjuk be a RUN parancsot, s a B megkapja a 10 értéket. A számítógép maga jól tudja, hogy a második READ utasítással a lista második elemét kell beolvasnia. Ennek oka az, hogy a számítógép mindig megjegyzi a korábban beolvasott érték helyét, s így rögtön tudja melyik a következő. Ha most a C értékét akarjuk hozzárendelni, elég ha a programhoz egy harmadik READ parancsot fűzünk:

```
40 READ C
```

Ezzel tehát pontosan azt értük el, mint a korábbi demóprogrammal. Mindhárom változó megkapta a maga értékét.

A számítógép most beolvasta a lista legutolsó elemét is. Ha most arra kérjük a gépet, hogy olvasson be még egyet, a gép méltatlankodni fog, hiszen ő már beolvasta az összes meglévő adatot, többről már nem tud. Ha tehát beadjuk a:

```
50 READ D
```

programot, a programunk futtatása után a gép az "OUT OF DATA ERROR" hibajelzést fogja kiadni.

A D változóhoz már nincs érték rögzítve. Mihelyst azonban a DATA listát egy újabb adattal egészítjük ki, máris korrektül fog a programunk dolgozni.

Hogy ne kelljen állandóan kiírnunk a READ utasítást, az operációs rendszer lehetővé teszi nekünk (hasonlóan a PRINT, a NEXT utsításokhoz) hogy rövidítsünk. A programunk ekkor így fog kinézni:

```
10 DATA 5, 10, 15, 20
20 READ A, B, C, D
```

Sok olvasó most talán azt mondja, ez a módszer komplikáltabb mint az "=" jellel történő hozzárendelés. Nagyobb listák esetében azonban mindjárt érzékelni fogjuk a szükséges gépelés közötti különbséget. Ugyancsak érdekes az is, hogy mi a listánkat bármikor újraolvashatjuk. Ehhez csupán annyit kell a gépünkkel közölni, hogy azt ismét az elejétől kezdve dolgozza föl. Ezt a RESTORE parancs kiadásával érhetjük el. Mihelyst a számítógép erre a parancsra bukkan, a lista aktuális értékre irányuló mutatóját ismét annak legelejére irányítja, akárhol is tartózkodott. Ha például az E, F, G, H változóknak ugyanazokat az értékeket kívánjuk adni mint az előzőeknek, akkor a programunkat az alábbiak szerint kell kibővíteni:

```
10 DATA 5, 10, 15, 20
20 READ A, B, C, D
30 RESTORE
40 READ E, F, G, H
```

A RESTORE miatt nem kapunk hibajelzést a következő READ utasításnál (bár nincs több adatunk), hanem elérjük, hogy az újabb változók is a lista eredeti értékeit kapják.

Tömbváltozók és listák

Ha tömbváltozókkal dolgozunk, akkor a DATA, a READ és a RESTORE utasítások különösen fontosak lesznek. Ha például egy 50 elemű tömböt akarunk konstans értékekkel föltölteni, akkor az "=" jeles megoldás egyszerűen elfogadhatatlan lenne. A munka egy FOR...NEXT hurokkal gyerekjáték lesz:

```
5 DIM A (50)
10 FOR INDEX=1 TO 50
20 READ A(INDEX)
30 NEXT INDEX
40 END
```

50 DATA 5, 10, 20, 40...
60 DATA 120, 990, 1000

Ha a fönti programot áttanulmányozzuk, rögtön megláthatjuk hogy az INDEX számlálót egyúttal a tömbváltozó elemeinek azonosítására is használhatjuk. Ez egy apró trükk, amivel azt érjük el, hogy a tömbváltozó értékeit egyszerűen dolgozzuk föl.

Természetesen arra is módunk van, hogy adatlistának sztringeket adjunk meg. Mindezekkel az ismeretekkel véleményünk szerint máris sokat lehet kezdeni. Viszont ez még nem minden, s a most bemutatandó parancsokkal olyan lehetőségek birtokába juthatunk, amit esetleg nem is gondoltunk volna.

A tároló manipulálása BASIC-ből

Azt bizonyára mindenki tudja, hogy a C64-es 64 kbyte-nyi tárolóterületet igazgat. Minden egyes byte megfelel egy tárolócímnek, amelyek 0—255 közötti értékeket képesek fölvenni. A számítógép csupán ezeket az értékeket képes megérteni. Amikor például beadunk egy BASIC programot, a tárolóban nem a képernyőn megjelenő betűk lesznek, hanem a parancsoknak és értékeknek megfelelő kódszámok. Hogy azután ezt a számunkra értelmetlen kavalkádot a LIST paranccsal érthető kódrendszerben jelenítsük meg magunknak, a gép operációs rendszerébe beépített program lefordítja azt számunkra BASIC-re. Ez a rendszer teszi lehetővé nekünk, hogy a géppel viszonylag egyszerűen kommunikáljunk. Amikor a gépet bekapcsoljuk, ez a rendszer azonnal aktiválódik.

Az összes tárolócím meg van számozva, a számozás 0-tól indul és 65535-ig tart. Az adott tároló sorszámát nevezzük annak címének, amivel azután azt el lehet érni. Hogy a számítógép végrehajthassa azokat a feladatokat, amelyekre azt szánták, ezért a tárolóban bizonyos területeknek bizonyos feladataik vannak. A legalsó tartomány a 0 címtől kezdve 1023-ig a már említett operációs rendszer ,,tulajdona". Ott a munkához fontos értékeket tárolunk. Azt ,,zeropage"-nek, azaz nulláslapnak is nevezzük. Ezt követően találjuk 1024-től a képernyőtárolót, amelyben azok az adatok találhatók, amelyeket mi a képernyőn látunk.

Ezután állnak a BASIC programok, amelyeket a képernyőn keresztül a gépbe viszünk. A BASIC tároló a legnagyobb tároló, és az a programozó szabad rendelkezésére áll. Abban kb. 38000 byte hely van, és az a 2048-tól a 40959-es címig ér. A maradék területet ismét az operációs rendszer és a BASIC interpreter foglalja el.

Különleges BASIC parancsok segítségével azonban célirányosan elérhetjük a tárolókat, illetve azok tartalmát meg is tudjuk változtatni. Az első parancs a POKE utasítás, amellyel a megadott címre 0—255 közé eső értékeket lehet írni. Csupán a kérdéses cím és a kívánt érték megadására van szükség:

POKE cím, érték

Ha például a 20000-res címre a 45-ös értéket akarjuk vinni, az alábbi parancsot kell bedni:

POKE 20000, 45

Szemmel láthatólag semmi sem történik. Pontosabban látható hatást ugyanúgy nem érzékelünk, mint a változók hozzárendelésekor sem. A paranccsal csupán azt értük el, hogy a 20000-res címre írtuk a 45-ös értékeket. Hogy ez valóban megtörtént, egy másik paranccsal tudjuk ellenőrizni. Ez a parancs a PEEK utsítás. Ha tehát ellenőrizni akarjuk a hatást, adjuk be az alábbi sort:

PRINT PEEK (20000)

A számítógép azonnal kiadja a 45-öt, hiszen mi ezt az értéket ,,pókoljuk" oda. Ezt az értéket akár egy változóhoz is hozzárendelhetjük:

A = PEEK (20000)

A 45-nek a 20000-res címre történő beírásával látható változást nem okoztunk. A tárolócímek tartalmának megváltoztatása akkor lesz igazán vonzó, ha olyan címekhez nyúlunk hozzá, amelyek szerepet kapnak a számítógép ,,életében" is.

Egyéb hatású POKE-ok

Lehetséges például a képernyő keretének és a lapszínnek a megváltoztatása. A POKE 53280,X a keret, a POKE 53281,X a lapszínt változtatja meg. A POKE 646,X pedig a tollunk színét alakítja. Az X a kívánt szín értékének felel meg, amely 0—15 közé eshet. A szín-kód megfeleltetést a gépkönyvből olvashatjuk ki.

*

Az előbb beszéltünk a nulláslapról. Ha itt változtatunk meg értékeket, annak esetleg ,,végzetes" hatása lesz. A C64-es kézikönyvében a Q függelékben találhatjuk a nulláslap címeit és azok jelentését. A legtöbbhöz valóban nem tanácsos nyúlni, hisz' azzal szépen belekotornánk a gép operációs rendszerébe. Ettől függetlenül nem kell félnünk a ,,kísérletezéstől", ugyanis legföljebb azt érjük el, hogy a gép lemerevedik és nekünk ki kell váltanunk az úgynevezett ,,Master-RESET"-et, ami annyit tesz, hogyha tovább akarunk dolgozni, ki majd újra be kell kapcsolnunk a gépet. Ellentétben tehát némely híreszteléssel, nem létezik olyan ,,killer POKE", amely a számítógépet hardveresen használhatatlanná tenné. Némely cím ismerete és POKE-kal történő befolyásolása pedig kimondottan hasznos dolog is.

Például esetleg zavar minket, hogy egy program futtatásakor a kurzor nem villog. A POKE 204,0 paranccsal, amelyet a programunk elején adhatunk be a kurzort ,,rendes" üzemmódba kapcsolhatjuk. Például ha a mellékelt programocskát beadjuk, villogni fog a kurzor, egyébként (a POKE nélkül) nem:

10 POKE 204, 0
20 GOTO 20

A RUN beadása után a programot a 20 sorban egy végtelen ciklusra állítjuk, a kurzor azonban úgy villog, mint a parancsmódban. A kikapcsolást is elérhetjük:

POKE 204, 1

Épp ilyen érdekes például a 650-es cím. Azzal lehet a billentyűk ismétlő funkcióját beállítani. Ha azt akarjuk, hogy a gép összes billentyűje ismételjen, akkor az alábbi parancsot kell beadni:

POKE 650, 128

Ha az ismétlés frekvenciáját is meg akarjuk változtatni, akkor egy olyan címet kell megváltoztatnunk, ami ugyan nem a nulláslapon van, mégis fontos lehet a számunkra:

POKE 56325, 1

Ha mindkét parancsot egyszerre adjuk be, nem valószínű hogy képesek leszünk a géppel dolgozni. Az 1-es érték felel meg a leggyorsabb, a 255-ös a leglassabb lekérdezésnek.

A BASIC-en kívül a gépünk még egy más nyelvet is megért, mégpedig a gépi kódú nyelvet. Erről talán már hallottunk is. A gépi kódban írt programok sokkal gyorsabban futnak, mint a BASIC-ben írtak. Ez a nyelv olyan, hogy azt a számítógép közvetlenül megérti, nincs szükség egy ,,értelmező" programra. A gépi kódban történő programozáshoz azonban kell egy kis tapasztalat és gyakorlat is, ezért a kezdők számára még nem ajánlható. Mindettől függetlenül semmi sem akadályozhat meg minket abban, hogy kihasználjuk mindazokat a rutinokat, amelyeket már beírtak a gép operációs rendszerébe. Most már csak az a kérdés, hogyan lehet egy ilyen rutint elindítani?

Mialatt mi egy BASIC programot a RUN paranccsal indítunk, a gépi kódú

programokat a SYS utsítás kiadásával kelthetjük életre. Csupán ez a parancs azonban nem elég, meg kell adnunk azt a címet is, ahol a rutin kezdődik. A BASIC programok a normál esetben mindig ugyanazon a címen kezdődnek (#0801), a gépi kódú programok azonban a még szabad területen bárhol állhatnak. Ezért kell az említett indítási cím. A parancs ekkor így fog kinézni:

```
SYS kezdőcím
```

Ha ismerjük a megfelelő kezdőcímet, máris fölhívhatunk operációs rendszer alprogramokat. A legutóbbi cikkben, ahol a PRINT utasításról is szót ejtettünk, már használtunk is egy ilyen rutint:

```
10 POKE 781, sor
20 POKE 782, oszlop
30 SYS 65520
40 PRINT "TESZT"
```

Ha bepókoljuk a megfelelő sor/oszlopszámot a megfelelő címre, a kurzort az adott pozícióra tudjuk irányítani. Előbb persze meg kell hívnunk a megfelelő operációs rendszerrutint, amit a 30. sorban láthatunk. A rutin kezdőcíme a 65520. Ezen kívül sok egyéb ilyen hasznos rutin található. Például ha szeretnénk a gépet a bekapcsolás utáni állapotba vinni, akkor ezt kell beadnunk:

```
SYS 64738
```

Ennek a parancsnak az érdekessége, hogy bár visszakapjuk a bekapcsoláskor megszokott feliratokat, de nem minden dolog ,,vész" el a számunkra, csupán a BASIC program, és az úgynevezett kazettapufferben álló gépi kódú programok.

Éppen ezért, de egyéb okokból is a SYS parancsokkal hasonló óvatossággal kell bánnunk, mint a POKE-kal, ugyanis könnyen lemerevíthetjük a gépünket.

A BASIC tanfolyamunknak mára vége. Úgy érezzük, hogy nagy lépéssel jutottunk tovább a megismerésben. Ezekután ha intenzíven foglalkozunk továbbra is a BASIC-kel, sokkal jobban elmélyedhetünk a C64-ben.

Még többet ésszel!

I. rész

Múlóban van a BASIC népszerűsége. Kimondatott rá az ítélet: gyermekeknek való, lassú és nem strukturált. Ezzel szemben a PC, C nyelven, de legalább Pascalban programozva, hát az aztán igen.

No kérem, ezennel elindítunk egy új cikksorozatot a technikás BASIC programozásáról, dacolva a divatáramlatokkal. Meggyőződésünk ugyanis, hogy a jó öreg BASIC hatékonysága semmivel sem rosszabb a többi magas szintű nyelvénél, csak kevésbé szoktak figyelmet fordítani a céltudatos, átgondolt programozására. Ezért mindenki hasznára terjeszteni kívánunk olyan módszereket, fogásokat, ötleteket, amelyekkel ügyesebb, hatékonyabb programokat lehet írni, kizárólag a BASIC keretein belül maradva. Szóba fog kerülni a programok gyorsításának lehetősége, a dokumentálás, és persze az a sokat emlegetett strukturálás is. Szeretnénk sokakban kedvet ébreszteni ahhoz, hogy továbbfejlesszék saját programozói képességeiket, kihasználva a BASIC számos jó tulajdonságát.

Cikkeink megírásához tapasztalt programozók segítségét kértük. A témát rendszerint a szerkesztőségnek küldött programok hibái, illetve kevésbé szerencsés megoldásai adják, de foglalkozunk egyéni ötletekkel és alapfogalmak magyarázatával is. Minden részben felteszünk majd egy kérdést, amelyre a válasz a következő számban lesz olvasható. A megfejtést teljesen felesleges beküldeni, gondolkozni rajta azonban érdemes lesz.

Mivel a Commodore-család tagjai közül a 64-es BASIC-je a legegyszerűbb, ezért a cikkben a bemutatott példák általában erre készülnek; ezek persze értelmezhetők más rendszerekre is. Igyekszünk kerülni a nagyon gépfüggő ötletek bemutatását, erre a Tippek és Trükkök rovat való. Munkára tehát, és mindenkinek kellemes okulást kívánunk!

a szerkesztő

Programot írni könnyű dolog, javítani azonban néha nagyon fáradságos lehet. Például azért, mert túl kusza a program gondolatmenete, logikátlan trükközéssel van tele, vagy csak egyszerűen nincs részekre tagolva. Azt hiszem, ezzel nem mondtam el semmi újat. De hát a programokat főleg írják és nem javítják, nem igaz?

Nagy tévedés! Ha csak a komolyabb, mondjuk, legalább harmincsoros programokat nézzük, akkor még egy olyat sem láttam, amelyik elsőre tökéletesen megfelelt az elképzelésnek. Legalább a kiíratás módján változtatni kellett, azaz javítani.

A javítás a program megírásának szerves része. Ilyenkor pontosan tudnunk kell, hogy a módosítás milyen hatással lesz a program többi részére. Ez pedig csak áttekinthető szerkezetű programban valósítható meg rendszeresen. Szerkezet latinul structura. Összegezve: ha nem akarok vakon belepancsolni a programomba az egyébként kikerülhetetlen javítgatáskor, akkor a programomat strukturáltan kell megírnom, megjegyezhetővé téve a magam számára annak felépítését. De mit is jelent ez a varázsszó?

Meg fogom magyarázni, de előbb tegyünk egy kis kerülőt és beszéljünk egy-két fontos apróságról. Többször fogok programokból idézni. Ezek szerzőit csak azért nem nevezem meg, mert nem őket akarom kipellengérezni, hanem a hibákat. Nem hiszem, hogy az, amit elmondok, az egyetlen és abszolút igazság. De tény, hogy sok-sok programot írtam már, és eközben nagyon hasznosnak bizonyultak azok az alapelvek, amelyeket megosztani szeretnék kevesebbet látott kollégáimmal. Ne sértsen senkit, ha esetleg szokatlan dolgokat ajánlok megfontolásra, mindig igyekszem érvekkel támogatni a javaslataimat.

Nézzük meg tehát az 1. példát, egy programból kiemelve. Szinte rögtön az elején van egy ugrás valahova nagyon hátulra, majd onnan vissza. Nem szeretem, ha a strukturált nyelvek elvakult hívei örökösen a GOTO utasításba kötnek bele, de itt most igazuk lenne. Ez a program ugyanis egy teljesen felesleges hurkot, kerülőt tartalmaz, amely csak nehezíti a gondolatmenet követését. Ha ugyanis a program elugrik valahová, nem látszik jól, hogy az utána következő részre (itt 150) honnan lép rá. És egyébként is, a végrehajtás során egymást követő részeket miért nem lehet egymás mögé írni? A számítógép is így hajtja végre, tehát így logikus.

Ezzel megközelítettük a strukturálás egyik alapelvét: egy program legyen felülről lefelé követhető. Egy strukturált (vagyis

jól megszerkesztett) program egymás után (műszóval szekvenciában) elhelyezett elemi részek sorozata. Háromfajta elemi részből felépíthető minden program szinte minden nyelven: elemi utasításokból, eljáráshívásokból és elágazásokból. De nem akarom ezeket a fogalmakat most részletesen körülírni, hagyjuk az újdonságot kissé megpihenni, és foglalkozzunk gyakorlatiasabb problémákkal. Például hogy miért jó, ha egy programot ilyen szabályok szerint, vagy csak egyszerűen az olvashatóságra ügyelve írunk meg.

Egy programot nem egy-két nap alatt szokás elkészíteni. Ha pedig egy olyan részt kell javítanunk — javítani mindig kell, megbeszéltük —, amelyet két héttel (vagy éppen hónapokkal) azelőtt írtunk. Ha elolvasva nem értjük meg biztosan saját művünket, akkor nem tudjuk javítani sem. Ehhez szükséges, hogy íráskor olyan szabályokat tartsunk be, amelyekre évek múltán is emlékezni tudunk. Ilyen szabályokat mindenki kialakíthat magának, egyéni szerkezeti megoldásokkal, jelölésekkel, egyebekkel. Megfigyelhető, hogy a főbb szabályok a gyakorlat növekedtével egyre jobban hasonlítanak mások szabályaira, és nagyjából ezek a kristályosodott elvek jelentik a strukturált programírás módszerét.

Előbb-utóbb mindenkivel megtörténik az is, hogy másoknak is megmutatja készülő programját, esetleg tanácsra, segítségre számítva. Ha ez a program a többiek számára nehezen olvasható, például mert a szerkezete nem követi az általánosan kialakult gyakorlatot, akkor a segítő szándékot könnyen felemésztheti a puszta olvasás is. És persze pórul járhatunk mi is, ha más rendetlen programját szeretnénk legalább megérteni. A következő számban elmesélek három olyan szabályt is, amelyeket az olvashatóság érdekében érdemes betartani. Most azonban térjünk vissza példánkra, és próbáljuk kitalálni, hogy mi szükség lehet egy ilyen ugrándozásra.

A BASIC nyelv egyik jellegzetessége, hogy minden sor számozva van. Ha az 1. példában 870-nél kezdődő, távolabbi részt később írjuk meg, akkor lehetséges, hogy a legjobb szándékkal sem fér el oda, ahova elvileg való, tehát a 150. sor elé. Ettől azért még nem kell rögtön a GOTO-hoz nyúlni. Nagyon elegáns dolog lehet, ha az új részt elhelyezzük valahol a program végén, és a GOSUB utasítással ugrunk ki rá (ld. 2. példa). Ekkor ugyanis egy RETURN utasítás hatására a program a szubrutin-hívás után folytatódik. Ez a végeredmény szempontjából majdnem tökéletesen megegyezik az előző változattal, csak most már első pillantásra is látszik, hogy mit akarunk. És ez a célunk, ugyebár.

Nem mindig a legcélszerűbb módszer szubrutinba írni az utólag felvett sorokat. A szubrutin akkor van ügyesen szerkesztve, ha pontosan egy logikailag összefüggő programrészt tartalmaz, amelyből ráadásul nem is lehet elugrándozni. Talán még nem tudja mindenki, hogy a program sorszámai megváltoztathatók, például a BASIC-ben néha megtalálható REN vagy RENUMBER paranccsal, vagy éppen a Help Plus (C64) #R paranccsal. Persze nem elég a sorok elején levő számok átírása, értelemszerűen új érték kell az összes hivatkozásba is (GOTO, GOSUB stb.). Vigyázat, a Simon's BASIC ezt elrontja! Tehát megfelelő átszámozással helyet lehet csinálni az új rutinnak, és így megtartható a logikus sorrend.

Sajnos mai ötpercünk eltelt, remélhetőleg nem hiába. Van két bemelegítő kérdésem a következő számhoz:

Milyen módon lehet a különböző ciklusokat elhagyni?

A strukturált program elemei között nem szerepelt a ciklus, mégsem tévedtem. Miért?

(1.)

```
130 ...
140 GOTO 870
150 ...

870 ...

940 GOTO 150
```

(2.)

```
130 ...
140 GOSUB 870
150 ...

870 ...

940 RETURN
```

Hódi Gyula

Hibaigazítás
(92/3., 4. oldal)

R,név,n,s$ a 'név' vektor n. elemének tartalmát az s$ stringváltozóba másolja. Az s$ helyett tömbváltozó is állhat. (S$=NEV$(N))

Áttérés COMMODORE-ról IBM gépre

VÁLTÓ

Mottó:
Minden gépnél van jobb

Segédprogramok, DOS SHELL

A számítógéppel rendelkezők, azzal rendszeresen dolgozók általában nem elégszenek meg az alaprendszer nyújtotta lehetőségekkel, hanem igyekeznek hasznos segédprogramokkal egyszerűbbé, hatékonyabbá tenni a gépük kezelését. Így van ez a Commodore gépeknél, gondoljunk csak a másolóprogramokra, gyorstöltőkre, monitorpgramokra, és így van az IBM gépeknél is. Itt csupán az az eltérés, hogy míg a Commodore gépeknél kevés olyan program van, amely a gép kezelésének minden, vagy majdnem minden részét támogatná, mint például a GEOS, az IBM gépeknél az ilyen jellegű segédprogramokból több is van, olyannyira, hogy a programok ezen csoportja már szinte külön „műfajjá" vált, DOS SHELL néven. Ezek a programok a DOS szinte minden részletét felölelik, kibővítik a lehetőségeket, s ráadásul még kényelmesebb is kezelni őket, mint a DOS-t. Ezek a programok általában nem menürendszerűek, a funkciók egyszerű billentyűzetkombinációkkal, vagy „rámutatásos" módszerrel elérhetők, és hosszabb-rövidebb magyarázó szöveggel, helppel is el vannak látva. Míg a Commodore gépeknél a felhasználó általában az alaprendszerrel dolgozik, itt inkább az azt kiváltó segédprogram használata a jellemző. Szeretnénk az alábbi, ismertebb DOS SHELL-eket röviden bemutatni: PathMinder, PcTools, Norton Commander. Több is létezik még ezeken kívül, de azok vagy nem elég ismertek, vagy ahhoz az újabb, általában nagyobb gépekre készült csoporthoz tartoznak, ahol a programról, vagy programrendszerről már nehéz eldönteni, hogy ez most egy DOS SHELL, vagy egy új operációs rendszer…

PathMinder

Az „Útkezelő" a teljeskörű DOS SHELL-ek egyik első képviselője, ami nem jelenti azt, hogy teljesen elavult volna. A program lehetőséget ad az összes fontosabb feladat, programfuttatás, másolás, szerkesztés, alkönyvtárok, meghajtók közötti váltás egyszerű, menürendszerű végrehajtására. A program elindítása után a képernyő legfelső sorában láthatjuk a választható menüpontokat. Két kurzorról is beszélhetünk, amit talán parancskurzornak és programkurzornak nevezhetünk, bár valójában egyik sem az operációs rendszernél megszokott, hanem egy adott szó (menüpont vagy programsor) inverzben, illetve más színnel jelzése. A parancskurzor a legfelső sorban, a menüpontok választásánál látható, a programkurzorral pedig a directory-bejegyzések között közlekedhetünk. A választás a kurzorvezérlő nyilakkal és az enter leütésével, vagy pedig menüben a menüpont első betűjének lenyomásával lehetséges. A második sorban az esetleges almenüpontok, vagy a menüponthoz tartozó rövid magyarázat látható. Ha olyan menüpontot választunk, amelyben almenü is van, akkor a választás után az almenü kerül fel a felső sorba. Az ESC billentyűvel léphetünk vissza egy menüből a felette lévőbe. A képernyő többi részén látható információ a főmenü OPTION pontjából állítható az alábbiak szerint:

Az OPTION almenü VIEW pontjából három beállítást tehetünk. Ha a HELP-et kérjük, akkor a képernyő bal oldalán az aktuális lemez tartalmát vagy annak egy részét látjuk, a bejegyzések nevével, kiterjesztésével és méretével. A jobb oldalon egy HELP látható, amely röviden ismerteti, hogy az adott menüpontban milyen lehetőségeink vannak. Ha a parancskurzor a RUN menüponton áll, akkor a különböző billentyűk funkcióit láthatjuk. az OPTION—VIEW—STATUS esetén a bal oldal azonos az előzővel, a jobb oldalon pedig a rendszeridőt, dátumot, az aktuális lemez jellemzőit (név, méret, foglalt, szabad), a memória felosztását (összes, használt, szabad), és a jelenlegi felhasználható adatait láthatjuk. Ha az OPTION—VIEW—DATA/TIME kijelzést választjuk, akkor nincs kettéosztva a kép, az aktuális lemeztartalmat látjuk mint eddig a bal oldalon, kiegészítve a bejegyzés dátumával és idejével. A három lehetséges beállítást közvetlenül is kiválaszthatjuk az F1, F3, illetve F4 billentyű lenyomásával.

A főmenü menüpontjai

RUN — Ha a programkurzor EXE, COM, BAT, vagy BAS file-on áll, a program elindítása.

FILE — Állományokkal végezhető műveletek: másolás, kiírás, átnevezés, áthelyezés, törlés, gyilkolás, attributumállítás, titkosítás, titkosítás feloldása.

A több file-on végezhető műveletek esetén a [space] billentyűvel jelölhetjük ki a kívánt bejegyzéseket. A gyilkolás a törlés „igazi" módja, ilyenkor nemcsak a bejegyzést törli, hanem a tartalmat is. A titkosítás átkódolja egy általunk megadott szó, mondat vagy bármilyen betűcsoport alapján a kívánt file-t, így azokat más nem tudja használni.

DIRECTORY — A katalógusokkal végezhető műveletek: Megnyitás (belépés az alkönyvtárba), lezárás, rendezés (név, kiterjesztés, dátum, méret szerint), létrehozás és törlés. A megnyitás és becsukás közvetlenül is történhet, az INS illetve DEL billentyűkkel.

EDIT — A file szerkesztése: beépített szövegszerkesztő. Ha katalógusnéven állva adjuk ki, új file létrehozása és szerkesztése.

OPTION — Beállítások megváltoztatása: aktuális meghajtó, választás képernyőkép közül, hiba esetén hangjelzés, a képernyőkép beállításai, a szerkesztőkép beállítása, szünet a programok lefutása után, alkalmazói menü módosítása, aktuális beállítások listája, aktuális beállítás kimentése.

APPLICATION — alkalmazói menü, általunk definiált menüpontokkal.

COMPOSE — DOS parancs közvetlen kiadása.

LOG — a PathMinder futás közben — ha ezt a menüpontot YES-re állítjuk, —

jegyzi, hogy ki dolgozik a géppel, sőt a kiadott parancsokat is, tehát hogy mit csinált a felhasználó. Ebben a menüpontban állíthatjuk be, hogy kérjük-e ezt a lehetőséget (YES—NO), a felhasználó nevét (USER), valamint azonosítóját (,,számláját", ACCOUNT). Ez utóbbi tíz különböző lehet, személyes (PERSONAL) vagy üzleti (1-től 9-ig). Ha kértük ezt a menüpontot, a PathMinder készít egy PM.LOG file-t, amiben tárolja ,,fekete doboz"-ként a történteket. Ez a file tömör formátumú, ha megnézzük, semmi igazán értelmest nem látunk benne. Ha olvashatóvá akarjuk tenni, futtassuk le a PathMinder részeként kapott LOG2TXT.EXE programot a következőképpen:

LOG2TXT PM.LOG LISTA.TXT

Most már láthatjuk például a TYPE LISTA.TXT, vagy a PM-ből FILE TYPE, hogy ki és mikor lépett be (LOGIN), illetve ki (LOGOUT), valamint azt is, hogy mit használt, milyen parancsokat adott ki. Természetesen csak a bejegyzésekre vonatkozó műveleteket tárolja, ahol programfuttatás, másolás stb. történt.

XDOS — kilépés a programból, visszatérés a DOS-ba.

Lengyel István

BASIC bővítések áttekintése

A legtöbb BASIC nyelvjárásban/bővítésben a kulcsszavakat 1 byte-os tokenekben tárolják le. A mellékelt kis program hozzáférhetővé teszi számunkra ezeknek a tokeneknek a teljes listáját, ha azt a felhasznált nyelvjárás vagy bővítés alatt futtatjuk le. A tokenek értékei $80 és $FF közé esnek, azaz 128 és 255 közé. A program tehát előbb sorokat készít ezekkel a számokkal és az adott hexadecimális byte-okkal, majd bepókolja oda a tokeneket.

Legvégül a főprogram törli magát (ezért ajánlatos azt az első futtatás előtt valamilyen adathordozóra kimenteni), és ezután a BASIC tárolóban listázhatóan megkapjuk a tokeneknek és azok értelmezéseinek listáját. Némely BASIC nyelvjárásban a gép a 204-es ($CC) kódnál kiakadhat. Ha kitöröljük a 204-es sort, úgy ezt elkerülhetjük.

A ,,TOKENFINDER" hasznos lehet olyan kulcsszavak keresésénél, amelyek a kézikönyvben nem szerepelnek. A profi még olyan dialektusokhoz is illesztheti a programot, amely két-byte-tokeneket használ (pl. a Simons BASIC).

A megadott sorszámokat ne változtassuk meg, mivel a rutin pontosan erre van beállítva. Az automatikus törlőrutin a 600-as sorban áll, ezt a sort úgy tudjuk beadni, ha a parancsokat a rövidítésekkel adjuk be (?=PRINT, pE=PEEK stb.).

Programlista: A TOKENFINDER kijelzi egy BASIC bővítés kulcsszavait

```
400 POKE 2, 127410 Z=PEEK(2)+1
420 IF Z=256 THEN550
430 H=INT(Z/16)
440 L=Z−16*H
450 IF H)9 THEN H=H+7
460 IF L)9 THEN L=L+7
470 H$=CHR$(48+H)
480 L$=CHR$(48+L)
490 PRINT CHR$(147);Z;"...";H$;L$;"...*"
500 PRINT"RUN 410"
510 POKE 2,Z
520 POKE 631,19:POKE 632,13
530 POKE 633,13:POKE 198,3
540 END
550 A=PEEK(44)*256+PEEK(43)+12
560 FOR I=0 TO 127
570 POKE A+14*I,I+128
580 NEXT I
590 POKE 2,29:PRINT CHR$(147)
600 Z=PEEK(2)+1:PRINTCHR$(19)10*Z:PRINT"rU600":POKE631,19:
    POKE632,13:POKE633,13:POKE198,3:POKE2,Z:END
```

Hibacsatorna lekérdezés

Ha a 1541-gyel, vagy bármilyen más Commodore floppyval dolgozunk, gyakorta jelentkezik valamilyen hiba. Ennek jele, hogy a lemezegységen egy dióda villog. Ha meg kívánjuk tudni, milyen üzenetet is küldött a floppy, problémákba ütközhetünk. Például hogy nem tudunk pótsorokat szúrni az adott programhoz, mert az túl hosszú vagy éppen listázás ellen védett. Vagy egy gépi kódú programról van szó. Azt mindenki tudja, hogy az INPUT utasítás, amelyre a kiolvasáshoz szükségünk van, parancsmódban (tehát sorszámok nélkül) nem használható. Alkalmazásakor az ILLEGAL DIRECT ERROR hibajelzést kapjuk. Ebben az esetben át kell ,,verni" a C64-est. Először is POKE 58,1-gyel azt hitetjük el vele, hogy programmódban vagyunk. A sor elhagyása után azonban a programmód lekapcsolódik. Így rögtön ez után az utasítás után (tehát azzal egy sorban) az alábbi parancsokat kell még beadnunk:

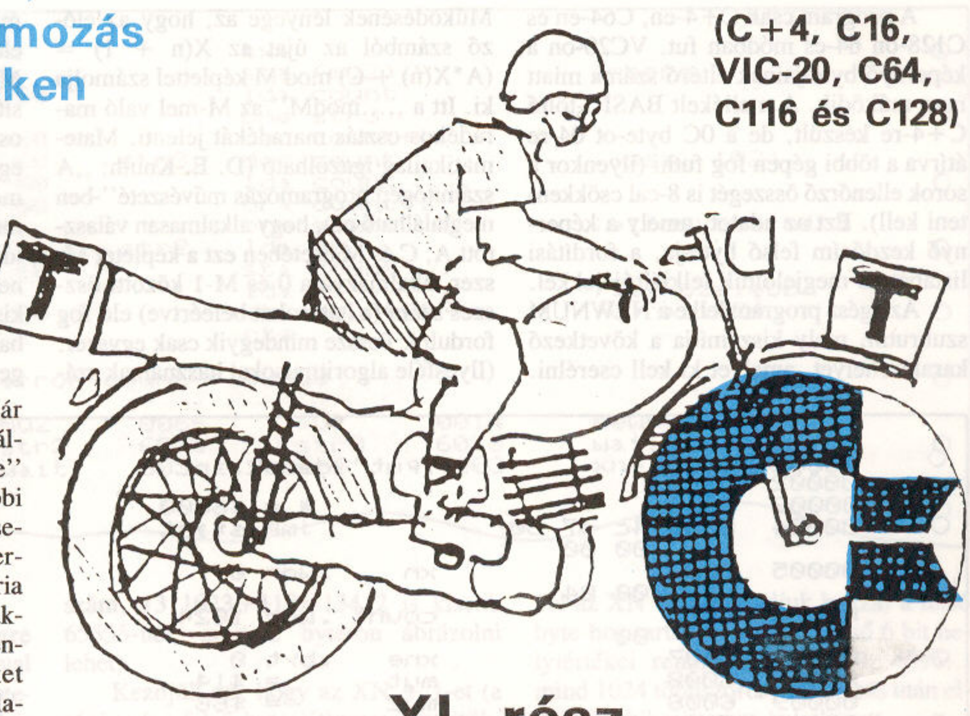
```
OPEN1,8,15:POKE58,1:FORI=1TO50:GET#1,A$:PRINTA$;
:CLOSE−(ST=64):IFST()64THENNEXT
```

Rendkívül érdekes az a trükk, amivel az összes parancsot egy sorba raktuk:

Egy IF után a C64 sajnos nem engedi meg az ELSE konstrukció használatát. Így tehát nincs alkalmunk lekérdezni azt, vég van-e az üzenetnek. Így vagy nem zárjuk be a csatornát, vagy mindig ötven karaktert olvasunk be. Márpedig egyik sem igazán jó. Így a CLOSE utasításnak az IF lekérdezés előtt(!) kell megtörténnie. Bezárni még mielőtt olvastunk volna? Nos a dologban van egy trükk, amihez a legjobban az ST státuszváltozót használhatjuk. Az (ST=64) kifejezés értéke mindaddig nulla, amíg az ST értéke nem egyenlő 64-gyel. Ellenkező esetben az érték —1 lesz. Márpedig amíg az üzenet véget nem ér, az ST értéke nem lesz 64! Így az olvasás végén a zárójelek elé helyezett mínuszjel hatására a —1 plusz eggyé válik. Ez pontosan az a csatornaszám amellyel az olvasási műveletet megnyitottuk. Addig, amíg az üzenet olvasódik, az egysoros a CLOSE0-val a nemlétező nullás csatornát zárogatja le, ami minket hidegen hagy. Végül tehát az üzenetet is megkaptuk, a csatornát is lezártuk, és a CLOSE tényleg az IF előtt áll. Lényeges viszont, hogy a csatorna megnyitásánál ne a ,,szokásos" OPEN15,8,15-öt használjuk, hiszen a CLOSE —(—1) az 1-es csatornát zárja. Ha valami oknál nem mondhatunk le a 15-ös csatornáról, a CLOSE—(ST=64) helyett a CLOSE(—15*(ST=64)) utasítást használjuk.

Gépi kódú programozás Commodore gépeken

(C+4, C16, VIC-20, C64, C116 és C128)



XI. rész

Cserebere a képen

Íme a megígért grafikus effekt. Már láthatta a Tisztelt Olvasó ezen rutin változatát, amikor ,,Véletlenszerű képcsere'' néven megjelent a C=újság egy régebbi számában. Az itteni változat célja kicserélni az éppen látható karakteres képernyőt egy $6000-rel feljebbi memória területtel (amely lehet egy másik karakteres kép) úgy, mintha teljesen véletlenszerűen kiválasztott karaktereket cserélgetnénk ki, végül mégis hiánytalanul kicserélődik a két kép.

```
10000 DATA 4C,47,60,00,00,00,04,88,017F
10001 DATA AD,03,60,0A,18,6D,03,60,0202
10002 DATA 0A,2E,07,60,0A,2E,07,60,013E
10003 DATA 18,6D,03,60,8D,03,60,AD,0285
10004 DATA 07,60,6D,04,60,8D,04,60,0229
10005 DATA AD,03,60,18,69,71,8D,03,0292
10006 DATA 60,AD,04,60,69,00,29,03,0206
10007 DATA 8D,04,60,60,A2,02,A0,40,02D5
10008 DATA 88,D0,FD,CA,D0,F8,60,20,0567
10009 DATA 09,60,20,3C,60,AD,03,60,0234
10010 DATA 85,14,85,66,AD,04,60,09,029E
10011 DATA 0C,85,15,09,60,85,67,A0,029B
10012 DATA 00,B1,14,AA,B1,66,91,14,032B
10013 DATA 8A,91,66,AD,05,60,D0,03,0366
10014 DATA CE,06,60,CE,05,60,AD,05,0319
10015 DATA 60,0D,06,60,D0,C9,09,00,0315
10016 DATA 8D,05,60,A9,04,8D,06,60,0292
10017 DATA 60,00,00,00,00,00,00,00,0060
20000 AS$="0123456789ABCDEF"
20001 DIMH(ASC("F"))
20002 FORI=0TO15:H(ASC(MID$(AS$,I+1)))=I
20003 NEXT
20004 BA=6*4096
20005 PRINT"KONTROLL INDUL !"
20006 FORI=0TO17:C=0
20007 FORB=0TO7
20008 READA$:GOSUB20029
20009 C=C+A
20010 NEXTB:PRINT".";
20011 READB$:A$=LEFT$(B$,2):GOSUB20029
20012 B=A:A$=RIGHT$(B$,2):GOSUB20029
20013 IFB*256+A<>CTHENPRINT"ADATHIBA A"10000+I"-EDIK SORBAN !!!":STOP
20014 NEXTI
20015 PRINT:PRINT"KONTROLL KESZ !"
20017 RESTORE
20018 FORI=0TO17
20019 FORB=0TO7
20020 READA$:GOSUB20029
20021 POKEBA+I*8+B,A
20022 NEXTB:PRINT".";
20023 READA$
20024 NEXTI
20025 PRINT:PRINT"OK."
20026 PRINT"INDITASA :"
20027 PRINT"SYS"BA
20028 END
20029 H=H(ASC(A$)):L=H(ASC(RIGHT$(A$,1)))
20030 A=H*16+L:RETURN
```

A program csak C+4-en, C64-en és C128-on 64-es módban fut. VC20-on a képernyő byte-jainak eltérő száma miatt nem működik. A mellékelt BASIC-töltő C+4-re készült, de a 0C byte-ot 04-re átírva a többi gépen fog futni (ilyenkor a sorok ellenőrző összegét is 8-cal csökkenteni kell). Ezt az adatot, amely a képernyő kezdőcím felső byte-ja, a fordítási listában is megjelöltük felkiáltójelekkel.

Az egész program lelke a NEWNUM szubrutin, mely kiszámítja a következő karakterhelyet, amelyet ki kell cserélni. Működésének lényege az, hogy az előző számból az újat az X(n + 1) = (A*X(n) + C) mod M képlettel számolja ki. Itt a ,,...modM'' az M-mel való maradékos osztás maradékát jelenti. Matematikailag igazolható (D. E. Knuth: ,,A számítógép programozás művészete''-ben megtalálható ez), hogy alkalmasan választott A, C és M esetében ezt a képletet M-szer alkalmazva a 0 és M-1 közötti összes szám (a határokat beleértve) elő fog fordulni. Persze mindegyik csak egyszer. (Ilyesféle algoritmusokat használnak számoló- és számítógépekben az egyenletes eloszlású véletlen számok generálásakor.) Minket itt ennek a képletnek a megvalósítása, a szorzás, összeadás és maradékos osztás érdekel. A program többi része egyszerű, a báziscím + véletlenszám címen lévő és a tőle $6000-re lévő byte-ot megcseréljük, várunk egy kicsit (itt az időzítés konstansai hasra beütött értékek, nekünk ennyi tetszett, lehet gyorsítani kisebb számokkal, vagy lassítani nagyobbakkal) és ezt csináljuk 1024-szer, ha vége, visszatérünk BASIC-be.

```
0000                   ;put"@demo2.src
0000                   ;
0000                            * = $6000
6000 4c 47 60                   jmp start
     00 00
6003                   xn      .wor 0
     00 04
6005                   count   .wor 1024
     00
6007                   xne     .byt 0
6008                   mut      = $14
6008                   mu2      = $66
6008                   ;m=1024
6008                   ;c=113
6008                   ;a=13
6008 ad 03 60          newnum  lda xn
600b 0a                         asl a
600c 18                         clc
600d 6d 03 60                   adc xn
6010 0a                         asl a
6011 2e 07 60                   rol xne
6014 0a                         asl a
6015 2e 07 60                   rol xne
6018 18                         clc
6019 6d 03 60                   adc xn
601c 8d 03 60                   sta xn
601f ad 07 60                   lda xne
6022 6d 04 60                   adc xn+1          ;X(n+1)=13*X(n)
6025 8d 04 60                   sta xn+1
6028 ad 03 60                   lda xn
602b 18                         clc
602c 69 71                      adc #113          ; +113
602e 8d 03 60                   sta xn
6031 ad 04 60                   lda xn+1
6034 69 00                      adc #0
6036 29 03                      and #3            ; MOD1024
6038 8d 04 60                   sta xn+1
603b 60                         rts
603c                   ;
603c a2 02             wait     ldx #2            ;ez pont
603e a0 40             wai1     ldy #64           ; ennyi idé..
6040 88                wai2     dey
6041 d0 fd                      bne wai2
6043 ca                         dex
6044 d0 f8                      bne wai1
6046 60                         rts
6047                   ;
6047                   start    * = *
6047 20 08 60                   jsr newnum        ;végy egy számot
604a 20 3c 60                   jsr wait          ;vár egy kicsit
604d ad 03 60                   lda xn
6050 85 14                      sta mut
6052 85 66                      sta mu2
6054 ad 04 60                   lda xn+1          ;képernyébeli
6057 09 0c                      ora #$0c          ;!!!!!
6059 85 15                      sta mut+1         ;pozició
605b 09 60                      ora #$60          ;másik kép
605d 85 67                      sta mu2+1         ;$6000-rel
605f a0 00                      ldy #0            ;messzebb lesz
6061 b1 14                      lda (mut),y
6063 aa                         tax
6064 b1 66                      lda (mu2),y
6066 91 14                      sta (mut),y
```

```
00061  6068 8a                txa
00062  6069 91 66             sta (mu2),y          ;csere
00063  606b ad 05 60          lda count
00064  606e d0 03             bne str3
00065  6070 ce 06 60          dec count+1
00066  6073 ce 05 60   str3   dec count            ;össz. 1024
00067  6076 ad 05 60          lda count
00068  6079 0d 06 60          ora count+1
00069  607c d0 c9             bne start            ;elfogyott már?
00070  607e a9 00      str2   lda #0
00071  6080 8d 05 60          sta count
00072  6083 a9 04             lda #4               ;legközelebb
00073  6085 8d 06 60          sta count+1          ;újra 1024
00074  6088 60                rts
end of assembly, error count = 00000

count    6005    mu2     0066    mut    0014    newnum   6008
start    6047    str2    607e    str3   6073    wai1     603e
wai2     6040    wait    603c    xn     6003    xne      6007
```

Nézzük azt a képletet. 13 = (1+2)*4+1. Ezt érdemes észre venni. Kettővel, vagy kettőhatvánnyal könnyű szorozni, hiszen az csak bit-léptetést jelent. Ezután a célszerű sorrend a következő, másoljuk le a számot (LDA) a másolatot léptessük balra (*2), adjuk hozzá az eredeti számot (2×+1×=3×), ezt léptessük kétszer balra (*4), majd adjuk hozzá ismét az eredetit. No ez meg van. Ha a C-t hozzáadjuk (113) és AND-eljük 1023-mal, meg is van az új szám.

Ez így elméletben rendben van, de a példa programban elég lazán kezeljük az adatokat, néha egy byte-ot léptetünk, néha két byte-ot adunk össze, amit nem is léptettünk..., talán nézzük meg korrekt-e ez. Világos, hogy 0—1023 közötti számokat két byte-on lehet tárolni, tehát ha két byte-tal számoljuk végig, akkor korrekt eredményt kapunk (a legnagyobb szám, 13*1023+113=13412 is kisebb 65535-nél, amit 2 byte-on ábrázolni lehet).

Kezdjük ott, hogy az XN + 1-et (a régi szám felső byte-ját) nem léptettük, tehát nem szoroztuk meg összesen 13-mal, hanem csak 1-gyel (nem változtattuk), így a szükséges eredménynél ⟨XN + 1⟩*256*12-vel kisebbet kaptunk (256-tal szorzás amiatt kell, mert ez a felső byte). 256*12=3072=3*1024. Ennek a maradéka a következő maradékos osztásnál (1024-gyel) úgyis nulla lenne, az eredményen nem változtat, jó hogy elhagytuk.

Ugyanígy elég ha az XN alsó 7 bitjével foglalkozunk (1024/13<(128). Továbbá nem viszünk hibát a számolásba akkor sem, ha az XNE változót nem nullázzuk a rutin előtt, pedig csak az alsó 2 bitjébe léptetünk új adatot. Az XNE (mivel az XN + 1-hez adjuk hozzá) a felső byte-hoz tartozik, ebből a felső 6 bit helyiértékei rendre 1024, 2048, 4096... mind 1024 többszörösei és osztás után eltűnnek, hiba itt sem keletkezett.

Persze ne gondoljuk, hogy mindent ilyen módon le lehet, vagy kell egyszerűsíteni. Gyakran nem is érdemes. Ezen rutin kezdete kb. két évvel ezelőtt íródott, és elkészülte után még fél évvel is volt mit elhagyni belőle. Előnye, hogy gyors és rövid, éppen itt céltalan, hiszen lassító ciklust használunk amúgy is. Azt a tapasztalatot mindenesetre levonhatjuk belőle, hogy gondosan eltervezve egyes kritikus rutinokat (amik gyakran futnak le) nagyon fel lehet gyorsítani programjainkat. Itt az első változat kb. kétszer ilyen hosszú és lassú volt (az mindenhol 16 bites léptetést és összeadást tartalmazott, a szorzás már volt csak ilyen).

Adatmentés

Előfordul, hogy a lemezen álló adatokat a floppy nem tudja beolvasni. Némelykor ez azért van, mert elállítódott a fej, esetleg a diszk erősen használt, vagy valami hasonló. Nos ilyen esetekben még próbálkozhatunk.

A floppy sokszor találkozik olvasási hibákkal. A beépített rutinok szerint ilyenkor újra megkísérli elvégezni a műveletet, ami a leggyakrabban sikerre vezet. A beállított próbálkozási szám az öt. A lemezegység RAM-jában a $69-es címen áll ez az 5-ös szám. Ha ide nagyobb értéket írunk, a kísérletek száma is növekszik. Esetleg ez segít a fontos adatok beolvasásában és új lemezre íratásában.

Persze a többszöri próbálkozás időbe telik, arról nem is szólva ha netán tíz, húsz helyen talál hibát a floppy. Mégis a tapasztalatok szerint sokszor megéri ez a módszer. Adjuk be tehát parancsmódban ezt a sort (X a kívánt próbálkozási szám):

```
OPEN 1,8,15,"M-W"+CHR$(105)+CHR$(0)+CHR$(X):CLOSE1
```

Megfejtés

Márciusi lapszámunk 23. oldalán közöltük olvasónk feladványát. A megoldást, íme most közzétesszük:

A hiba oka a PATH utasításban keresendő. RESETkor a PATH üres, tehát a vírusölőt CSAK a KILLED könyvtárban keresi a program, egyéb esetben pedig a már meglevő PATH-ot végignézi, és mivel a keresési útban benne van az A:\ és itt van a VIRUSOLO, megtalálja és elindítja. A hiba kiküszöbölésére csak be kell írni az indítás után egy olyan PATH-ot a VIRUSOLO meghívása elé, ami nem tartalmazza az A:\-t. Például elég egy PATH C:\

Szabó Balázs

SZÁMALK-CED
Kereskedelmi Kft.

Magas szintű irodatechnikai szolgáltatások a SZÁMALK-CED-től!

UTAX-Fénymásolók és teljes tartozékok (12-féle géptípus) — 39 900 Ft-tól + ÁFA

NIKKAM-pénztárgépek (APEH-engedélyes — visszaigényelhető) — 50 000 Ft-tól

SAFAX-telefaxok (francia) — 89 000 Ft-tól

KELLÉKANYAGOK:

— festékek, előhívók,
— faxpapírok,
— pénztárgépszalagok.

Tekintse meg bemutatótermünket!
Várjuk szeretettel!

Budapest XI., Budafoki út 109.
Tel./fax: 1810-757 vagy 1868-333/137

CÉL A CED!

Nyelvek Amigára

A programozási nyelveket nem szabad csupán egy jelölésrendszernek tekinteni. A nyelvet megértő software egészen átalakíthatja gépünket, a hardware korlátokon belül teljesen új tulajdonságokkal ruházza fel azt.

A matematikai nyelvek gondolata jóval megelőzte a számítógépek megjelenését. Már a század elején kísérleteztek egy, a matematikai problémákat jobban leíró „nyelvezet" kialakításán, de ezek az úttörő próbálkozások sajnos mind kudarcba fulladtak.

Az első, most már programozási nyelvet Konrad Zuse német matematikus 1945-ben doltozta ki. Nyelvét Plankalkülnek nevezte, és a hírhedt V2-es rakéták pályájának optimalizálására szolgált. Gépen sohasem alkalmazták, csak papíron, a gondolatmenet lejegyzésében segített. Ennek ellenére a mai nyelvek ősének lehet tekinteni, amelyből sok ma is használatos nyelv merített elemeket.

A Fortran

A legöregebb, Amigán is elterjedt nyelv a Fortran (Absoft AC Fortran — 475 DM /létezik egy 68020-as processzorra adaptált verziója is — 998 DM/). Ezt a nyelvet John Backus az IBM dolgozója fejlesztette ki 1945 után tudományos számítási műveletek elvégzésére. Nevét a FORmula TRANslation (képletfordítás)-ból kapta. A maga korában átütő sikernek számított, mivel sikerült olyan fordítóprogramot készíteni hozzá, amelynek kimeneti teljesítménye megközelítette a kézi kódolású programokét. A Fortran még sokat merített Zuse rendszeréből, és annak ellenére, hogy fejlesztését 1957-ben leállították, valószínű, hogy ennek a nyelvnek volt és van a legnagyobb befolyása a későbbi eredményekre.

A Lisp

Szintén régi nyelvnek lehet tekinteni, a még ma is alkalmazott, a nagyon magas szintű nyelvekhez tartozó Lisp nyelvet. Az 1958-ban elindult, John McCarthy (MIT) által vezetett fejlesztési munkák egy függvénykifejezések kiértékelésén alapuló nyelvre irányultak, így született meg a LISt Processing languange (listafeldolgozó nyelv).

A 70-es évekig ez a nyelv volt az egyetlen az úgynevezett nem Neumann-elvű nyelvek képviselője, amelyek köre mára már a mesterséges intelligenciakutatások miatt jócskán kibővült, így tagjai közé sorolhatjuk például a Prologot is.

Ezeket a típusú nyelveket nem a numerikus számítások igényeire, hanem szimbólikus, matematikai kifejezések formális differenciálására tervezték.

Az Amiga változatot a MetaComCo cég készítette.

A Basic

Bizonyára sok olvasónk felsóhajtott magában, amikor meglátta a Basic címet. Annyiban igazuk van, hogy profi alkalmazói körökben ez a nyelv a legfeketébb bárány, de két okból mégis foglalkozni kell vele. Az első, hogy a géphez alapkiépítésben adott software-ek egyike egy Microsoft bázisú Basic nyelv, a második, hogy az Amigán az assembly nyelven kívül Basic fordítókból és értelmezőkből van a legszélesebb kínálat.

Az Amiga legismertebb Basic nyelve, az AmigaBasic 1.2, amely igencsak körülményesen kezelhető, és szerintem minden próbálkozót megizzasztot.

A Basic kedvelőinek az első figyelemreméltó alkotást az Absoft cég dobta piacra AC Basic Compiler (275 DM) néven, amely az ASCI formátumban elmentett programokat fordítja be többkevesebb sikerrel és eredménnyel.

A második, már érdekesebb programmal a más gépeken igen jó hírnévre szert tevő HiSoft jelentkezett. Ő szintén egy compilert készített (165 DM), amely már jóval felülmúlja az Absoft termékét.

A HiSoft után a GFA következett. Az ő szakemberei egy fordítót (89 DM) és egy értelmezőt (175 DM) is készítettek, amelyből az előbbi egyes felhasználók véleménye szerint „csodákra képes", az előállított tárgykódok sebessége megközelíti a C-fordítók átlagos teljesítményét.

A sort a True cég zárja saját Basic-jével és fordítójával, amely igazán csak minden kiegészítőjével együtt üzemel jól, de ekkor ára már az egekbe szökik.

Az itt megemlített értelmezőkön és fordítókon kívül léteznek másfajták is (pl.: Professional Basic, MetaComCo Basic, Blitz Basic — 298 DM), de ezek kis teljesítményük vagy csekély elterjedésük miatt nem érdemelnek említést.

A Pascal

A Pascal-t 1970 után Niklaus Wirth tervezte a zürichi Műszaki Egyetemen, hasonló feladatokra, mint a szomorú emlékű Basic-et, de később alkalmasnak találván számos más feladatra is alkalmazták (MetaComCo Pascal).

Bár a Pascal-t nem tisztelik igazán a programozói berkekben, azért nagyon fontos, mint az őt követő nyelvek elődje. Wirth például a 80-as évek közepére a Pascal-ból kiindulva elkészített egy új nyelvet, melyet Modula-2-nek keresztelt. (Amigán: Bechmark Modula-2).

A Modula-2 Amigára készült változatában az a szép, hogy megfelelő kiegészítésekkel elfogadja a C- és assembly forráskódokat is.

A Forth

A Forth nyelvet 1970 végén Charles H. Moore fejlesztette ki, az amerikai NRAD rádiócsillagászati obszervatórium munkatársa. Célja az volt, hogy a folyamatszabályozásra, különösen a távcsövek vezérlésére alkalmas rendszert teremtsen. A Forth azonban nem akadt le ezen a területen. Sok számítógépes alkalmazását megtalálhatjuk, amit rendkívüli tömörségének tudhatunk be. Egyébként ez a sűrűség ijeszti el a kezdőket e nyelv megtanulásától, ugyanis már a Forth forráskód is annyira tömör, hogy több kulcsszó puszta elválasztójel.

Amigán a Multi cég fordítója vált ismertté.

A C

A C viszonylag fiatal programozási nyelv. Eredetileg a Bell Laboratoriumban (Murray Hill, New Jersey) az UNIX operációs rendszer támogatására készítették. Tervezője Dennis Ritchie a BCPL nyelvből, annak kiszélesítésével fejlesztette ki, a Bell Labor PDP-7-es gépéhez készült B nyelv felváltására.

A B nyelvet egy bizonyos Ken Thompson írta, aki később az ismert Pascal nyelv megalkotásával szerzett hírnevet. A B már meglehetősen rugalmas volt a gépi környezettel szemben, programjait egyszerűen át lehetett írni egyik gépről a másikra, de még mindig nem felelt meg a profi követelményeknek.

A B nyelv hiányosságaira épülve készült el egy új nyelv, a C.

Sokan azt gondolják, hogy ennek a nyelvnek a neve egyenesen a B nyelv nevéből következik feltételezve azt, hogy létezik egy A nyelv is. Pedig tévednek, a BCPL nyelv első továbbfejlesztéséből született új nyelvet stílusosan — vagy fantázia hiányában — az első betű B-jéről nevezték el. Így az új fejlesztés eredménye az anyanyelv második betűjét, a C-t kapta.

Annak ellenére, hogy a C-t a BCPL-ből fejlesztették ki, nem sok közös tulajdonságot mutatnak. A BCPL-ben az egyetlen adattípus a gépi szó, addig a C-ben a lebegő-, pontos-, egész számok és karakterek is az alapvető objektumok közé tartoznak.

Gépközelsége miatt a tesztelések során a rendszerprogramozók kimagasló értékelést adtak a C nyelvről. Rendkívüli tömörsége, kis utasításkészlete assembly közelébe helyezte, ami nem jelenti azt, hogy egy alacsony szintű nyelvvel lenne dolgunk.

Talán ezért is kezdetben szinte kizárólag csak olyan szakemberek alkalmazták a C-t, akik járatosak voltak valamelyik rendszer assembly-ében, mivel a két nyelv hasonlatos utasításkészlete nem okozott számukra nagy problémát.

A későbbi fejlesztések alkalmával, amelyekben számtalan egyetem vett részt, sok eltérő rendszer kezdett elterjedni, melyek csak az utóbbi időben mutatnak egységes képet, de az egység mára már annyira valós, hogy túlszárnyalta a nemzetközi szabványok által meghatározott nyelveket, mint például a Fortrant vagy Cobolt. A C nyelv legújabb verziói már emberközelibbek és a közel magas szintű nyelvekhez sorolhatók. Ezek már nem hasonlítanak annyira az assembly-re, de megőrizték annak hatékonyságát és rugalmasságát, megkímélve a felhasználókat a hatalmas erőfeszítésektől, eltüntetve a magas szintű nyelvek korlátait.

A C általános célú programnyelv. Hiba lenne rendszerfejlesztő nyelvként szólni róla (mint ahogy sokan teszik) — bár megjelenése előtt sohasem írtak operációs rendszert assembly-nél magasabb szintű nyelven —, mikor valójában széles körben alkalmazható, a mai számítógépek tudását tükröző, kifejezőképes nyelv, amely elég hatékony ahhoz, hogy megkíméljen minket az assembly fáradalmaitól.

A C tartalmazza a jól felépített programok készítéséhez szükséges alapvető vezérlési szerkezeteket, lehetővé téve a címaritmetikát, a függvények és mutatók alkalmazását.

Alapállapotban nem tesz lehetővé multiprogramozást, csak egy szálon futó vezérlésátadási struktúrákat tartalmaz, ami hibának tűnhet, de ez okozza a nagymérvű kompatibilitását, a gépfüggetlen programozást.

Sokan úgy vélik, hogy a C nem igazán szerencsés az amatőr programozók számára. Erről el lehetne vitatkozni. Sokunk álláspontja szerint a ,,Basic-nél minden jobb" kijelentés több nyelv mellett szólhat, viszont ezek közül a C nagyfokú tisztaságával emelkedik ki. Használata során nem rögződnek olyan káros programozói szokások, amelyekre a Basic, Pascal és sok magasszintű nyelv kényszeríti alkalmazóit.

A C nyelv ellenzői mellett szól az interpreter hiánya, ami megnöveli a programozásra fordítandó időt, megkeserítve a felhasználók életét.

A jövőben a C nyelv ugrásszerű fejlődésére lehet számítani, amelyet az UNIX rendszerek elterjedésének köszönhet. Az új verziókban talán már helyet kap egy értelmező is.

A fejlődés előfutára egy új nyelv a C++, amely túlhaladva a C-n egy új, még alulról kompatibilis nyelvi irányzatot indított útjára.

Ennyit bevezetőnek és reklámnak a különböző, Amigán elterjedt nyelvekről, melyből mindenki eldöntheti, hogy nekigyűrkőzzék-e egy új nyelv megtanulásában rejlő nehézségeknek.

Directory védelem

Egy jó directory védelem aranyat ér, még akkor is, ha ezt senki sem tudja megindokolni. Azután: minél egyszerűbb a védelem, annál sikeresebb is. Nos tároljunk le egy programot a lemezen "," néven. A betöltés máris problémás lesz. Hiszen a: LOAD ","8 nem fog menni. Marad hát a joker, azaz a LOAD ","8 vagy a LOAD "*",8 — ha a file az első a tartalomjegyzékben. De mit teszünk, ha ott csupa egybetűs állományok vannak? Akkor pedig jó szórakozást a vessző-állománnyal, amelyet se betölteni, se törölni, se átnevezni nem tudunk!

A szűrőprobléma megoldása

Igen sok program használja ki a C64-nél a gép kiváló zenei lehetőségeit. Sajnos azonban főleg ott vannak problémák, ahol igazán szép hatásokat lehet elérni. A problémás gyerek pedig a SID analóg szűrője. Ugyanis korántsem igaz, hogy az egyik C64 megegyezne a másikkal. Különbség lehet két kondenzátor méretezésében, pont azoknál, amelyek a szűrőért felelnek. A két kondi a C10-es és a C11-es. A régebbi gépeknél itt 470pF-os, az újabbaknál 2.2nF-os kondenzátor található. E téren egyébként a számos könyvben és egyéb helyen megtalálható kapcsolási rajz is eltér. A zavart az okozza, hogy a Commodore a C64 fejlesztésénél rosszul méretezte a szűrőt a 470pF-dal. Így ugyanis az a frekvencia tartomány, amelyben a szűrő működik, sokkal kisebb, mint amit a SID produkál. Így hát egy nagyobb érték lehet a megoldás (mondjuk 2.2nF). Ezért változtatta meg valószínűleg a Commodore a kapcsolást. Akinél a C10-es és a C11-es kondenzátorok 470pF kapacitásúak, könnyen beállíthatják az új ,,szabványt". Csupán annyit kell tenni, hogy ezekkel párhuzamosan be kell forrasztani egy másikat. Erre 1.5nF kapacitású kondenzátor kínálkozik. Összkapacitásként 1.97nF értéket kapunk, ami máris megközelíti a 2.2pF-ot. A kondenzátor fölépítése mindegy. Egy viszont fontos: Elkót (elekrolit kondenzátort) nem szabad használni! A legjobb ha syroflex vagy kerámia kondikat építünk be.

A beépítést az alábbiak szerint kell elvégezni:

Nyissuk föl a C64-et (vigyázat, a garancia elvész!). Ezután meg kell keresni a házban azt a panelt, amelyre C10 és C11 van írva. Most megnézhetjük a kondenzátorok feliratát, hogy azok 470nF vagy 2.2pF értékűek-e (legtöbbször valami 47 vagy 22 áll ott!). Ha az utóbbiról győződnénk meg, akkor semmit nem kell tenni, a gépet össze lehet szerelni. Ellenkező esetben beforraszthatjuk az 1.5nF kapacitású kondit. Ha ezzel készen vagyunk, ismét össze lehet szerelni a gépet. A legtöbb zenénél bottfüllel is észre fogjuk venni a különbséget.

Mindez abszolút rutinmunka, feltéve hogy tudunk bánni a pákával.

Pascal a C64-gyel

Aki azt állítja, a C64-es csak BASIC-ben és gépi kódban programozható, téved. Olyan nyelveken is gyakorolhatunk mint amilyen a PC-ken ismert C. Vagy épp a Pascal. Utóbbit például a Pascal-System segítségével tehetjük.

A C64-re írt Pascal-System két részre tagozódik, a könyvre és a programlemezre. A lemezen található a Pascal compiler, beleértve az editort és néhány példaprogramot. A könyv egy teljes Pascal bevezető tanfolyamot tartalmaz, illetve egy részletes kezelési leírást a compilerhez és az editorhoz. Mialatt a profik számára jónéhány tippet és trükköt is kínálnak, a kezdő a sok példaprogramon és a megadott feladatokon tesztelheti le a frissen szerzett ismereteit.

A Pascal-system

A Pascal compiler olyan forrásszöveg programokat tud feldolgozni, amelyeket a hozzáadott editorral készítettünk el. A szövegbevitel egy szövegablakon keresztül történik, amelyet a szövegen mind a négy irányba el lehet tolni. Az editor kimondottan komfortos, az a programozót szinte elkényezteti a sok hasznos funkcióval. A képernyő áttekinthetően tagolt, s a programozónak a forrásszöveg gyors és kényelmes felhasználását teszi lehetővé. Például az ⟨F1⟩ és az ⟨F3⟩ funkciós billentyűkkel az elkészített listát át lehet „lapozni". Ily módon az adott szövegrészek keresése nagyban leegyszerűsödik. Az olyan parancsok megléte, mint a FIND és a CHANGE, amelyek bizonyos szövegrészek keresésére és cseréjére valók, az editor szókészletében, magától értetődik.

A szöveg editálása kész szórakozás. A szerző az összes fontos editálási funkciót figyelembe vette az úgynevezett „LINE-Commands"-ban. Ezzel szavakat, sorokat vagy egész bekezdéseket is másolni, törölni vagy bármely irányba eltolni lehet.

A Pascal editoron belül csupán 8 kbyte szöveges tároló áll a rendelkezésre. A lemezen található programszövegek bezárásával azonban tetszőlegesen nagy programot tudunk moduláris felépítéssel létrehozni. Ehhez előbb ki kell fejleszteni a program egyes részeit, majd ezeket forrásszövegként a lemezre kell vinni. Ezeket a programokat a fordításnál össze lehet fűzni egyetlen egy programmá. Ha ennek a forrásprogramnak a compilálása után egy újabbat kell utántölteni és lefordítani, úgy csupán annyit kell tenni, hogy ennek az állománynak a nevét egy kommentársorban az első forrásszövegben meg kell adni. Például: (*$″FORRÁS 2.INC″*)

A compiler

A compiler menüben a felhasználónak három pont áll a rendelkezésére. A „CHECK SYNTAX" opcióval lehet átvizsgáltatni a forrásszöveget a szintaktikai hibákra. A „GENERATE CODE" opció ezután egy futtatható programot generál a BASIC tárolóterületen. Egy tetszőleges cím megadásával a tárgyprogramban lokalizálni lehet a hozzátartozó pozíciót a forrásszövegben. Az opciók kiadása a képernyőre történik, de a nyomtatón történő követésre is mód van.

A compilálási eljárás befejezése után elhagyjuk a Pascal-Systemet, és a kész programot RUN-nal elindíthatjuk. A tárgyprogram a BASIC tárolóban áll, s azt mint a hagyományos BASIC programokat tudjuk betölteni, indítani és lemezre menteni. Természetesen ilyenkor nem szabad BASIC sorokat toldani a programba, hisz' ez tönkre tenné a Pascal programunkat. A kész programot a Pascal-System nélkül is behívhatjuk a tárolóba és el is indíthatjuk. Az tehát teljesen független lesz a rendszertől, s bármely C64-esen futtatható. Az ílymódon elkészített programok általában háromszor-négyszer gyorsabbak mint az összehasonlítható BASIC programok.

A compiler elfogadja a Wirth-szabvány összes szavát, de ezen kívül rendelkezik néhány saját bővítéssel is. A parancskészletet a cikk végén megadjuk.

A Pascal programokban a kommentárokat általában a kapcsos zárójelek — {...} — közé helyezik. Mivel azonban a C64-es billentyűzetén nincsenek meg ezek a jelek, így a kommentárokat gömbölyű zárójelek és csillagok —(*...*) — közé kell tenni.

Ahogy az a többi compilernél is szokásos, a compilálási folyamatot „aktív kommentátorokkal" befolyásolni lehet. Ahogy azt a fönti példán a szövegek láncolásánál láthattuk, az ilyen kommentárokat egy dollárjel ⟨$⟩ vezeti be.

A dokumentáció

A könyvet három részre oszthatjuk:
— Pascal tanfolyam
— Tippek és trükkök
— Compiler leírás

A Pascal tanfolyam a könyv kb. felét teszi ki. Az tulajdonképpen bevezető minden Pascal-újonc számára, és főleg a mellékelt Pascal-Systemre orientálódik. A tanfolyamot igen sok hasznos példa is támogatja. Minden fejezet végén találunk az adott témával kapcsolatos feladatokat.

A könyv második része foglalkozik a Pascal-System részletes ismertetésével, bemutatásával. Az editor kézikönyve igen részletes, és nagyban megkönnyíti a programozónak a rendszer kezelésének elsajátítását.

A tippek és trükkök rész rövid bepillantást ad a Pascal nyelvnek az adatfeldolgozásban való alkalmazásához. A szerző hasznos ötleteket ad arra nézve, hogyan lehet a BASIC-ből már ismert SYS, POKE és PEEK hivatkozásokat a Pascal nyelvben is megvalósítani.

FACIT

A Florian Matthes által írt Pascal-System állja az összehasonlító próbát sokkal drágább Pascal programokkal is. A Pascal compiler, -tanfolyam és dokumentáció úgy a kezdőnek mint a profitnak megadja a jutányos lehetőséget, hogy megtegyék első lépéseiket a magasabbrendű Pascal nyelv világában. A német nyelvű könyvet a Markt & Technik Verlag 3-89090-222-7-es ISBN szám alatt forgalmazza.

A C64-re írt Pascal-System compilerjének parancskészlete:

AND	FUNCTION	RECORD
ARRAY	GOTO	REPEAT
BEGIN	IF	SET
CASE	IN	THEN
CONST	LABEL	TO
DIV	MOD	TYPE
DO	NIL	UNTIL
DOWNTO	NOT	VAR
ELSE	OF	WHILE
END	OR	WITH
FILE	PACKED	
FOR	PROCEDURE	
FORWARD	PROGRAM	

Egy más VAL

Leckéztessük meg a C64-est matekból! Mostantól a VAL parancs nemcsak a sztringek számmá való konvertálásának bugyuta feladatát végzi el csupán. Változókat, függvényeket és operátorokat is használhatunk ugyanis ezentúl. És mindezt Assembler sebességgel végezzük!

Kettő meg kettő mikor lesz kettő? Nos mindig, ha a VAL függvényt használjuk. Adjuk csak be: PRINT VAL("2+2") és győződjünk meg róla magunk. A VAL ugyanis nem igazán intelligens. A kiértékeléssel például azonnal leáll, mihelyt egy nem numerikus karakterre bukkan (itt a plusz jelre). Ezek miatt a korlátozások miatt a VAL az egyik legritkábban használatos függvény a parancskészletben.

Mindez azonban megváltoztatható. Az EVAL nevű programunk úgy módosítja a VAL-t, hogy az a sztringeket a BASIC-hez hasonlatos módon értékelje.

Másképpen mondva az X=INT((Y/2)+3) ezentúl ugyanazt az eredményt adja mint az X=VAL("INT((Y/2)+3)").

Mindez teljesen új lehetőségeket ad a BASIC-nek. A képletekkel dolgozó programok készítése gyerekjáték lesz, a függvényeket elegendő egyszerűen sztringváltozóba rakni majd mondjuk az Y=VAL(F$)-ral fölhívni stb.

Az EVAL program indítása után válasszuk meg a kezdőcímet. A legjobb ha elfogadjuk a megajánlott 49152-t a ⟨RETURN⟩ megnyomásával. Ebben az esetben az EVAL ezt a területet foglalja el. Ettől kezdve az új VAL él. A sztringekben most használhatjuk az alapműveleteket, a szög- és az exponenciális függvényeket, zárójeleket, változóneveket, logikai operátorokat stb. A VAL-nak átadott sztring kiértéelése azonos elvek és sorrend szerint történik mint a BASIC-ben pl. a PRINT után. Változókat is használhatunk! Pl. a **FOR I=1 TO 8;PRINT VAL("2↑I");:NEXT** eredménye az 1, 2, 4, 8, 16, 32, 64, 128-as sorozat lesz.

Természetesen az EVAL saját programjainkban is használható. Ehhez be kell hívni a programot, el kell indítani és ki kell választani a megfelelő startcímet. Az installálás után a BASIC betöltő törölhető és a saját program máris használható.

A VAL szintaxisa semmit nem változik. Így továbbra is meg kell felelnünk az általános szabályoknak. Ha hiba lépne föl, az EVAL nulla értéket ad, és egy hibaszámot rak le a 781-es cellába. Tehát nincs BASIC hibajelzés! (Ez igen jó a függvényplotter programok számára, amelyek nem merevednek le, ha az értékelési sorban egy, az értelmezési tartományon kívül eső érték szerepel. Pl.: **FOR X=–5 TO 5;PRINT VAL("1/X")**

A hibaszámok jelentése az alábbi:

```
 0 nincs hiba
11 SYNTAX ERROR
14 ILLEGAL QUANTITY ERROR
15 OVERFLOW ERROR
16 OUT OF MEMORY ERROR
18 BAD SUBSCRIPT ERROR
20 DIVISION BY ZERO ERROR
22 TYPE MISMATCH ERROR
23 STRING TOO LONG ERROR
25 FORMULA TOO COMPLEX ERROR
27 UNDEF'D FUNCTION ERROR
```

Természetesen ha akarjuk, nem kell tudomásul venni a hibaszámokat, ha elfogadjuk, hogy ilyen esetben az új VAL a nulla értéket adja át. Egy kiegészítést még tennünk kell, míg a BASIC a STRING TOO LONG hibajelzést a 255 karaktert túllépő sztringekre adja, az új VAL csak 88 karaktert tesz lehetővé.

Ha azt szeretnénk, hogy hiba esetében BASIC hibajelzést kapjunk, anélkül, hogy külön táblázatot kelljen a programunkba tenni, az alábbi utasítást írjuk a VAL után:

IF PEEK (781) THEN SYS 42039

Az EVAL nem használható olyan programokkal, amelyek használják az 1015–1023 ($03A7–$03FF) területet.

```
100 PRINT"ΞEVAL 1.0"
110 GOSUB 1000
120 PRINT"EVAL 1.0 MUKODESRE KESZ !"
130 PRINT"XPROBALD KI IGY: PRINT VAL("CHR$(34)"SQR(9)"CHR$(34)")
140 END
1000 PRINT "ΞEVAL 1.0
1010 INPUT"BETOLTESI CIM ";SA:CS=0
1015 PRINT "KIS TURELMET..."
1020 H1=INT((SA+13)/256):L1=(SA+13)-H1*256
1030 H2=INT((SA+150)/256):L2=(SA+150)-H2*256
1040 FOR AD=SA TO SA+206:READ ML:CS=CS+ML:POKE AD,ML:NEXT
1050 IF CS<>23175 THEN PRINT"HIBA AZ ADATSOROKBAN !":END
1060 POKE SA+5,L1:POKE SA+9,H1:POKE SA+133,L2:POKE SA+138,H2
1070 SYS SA
1080 RETURN
1090 DATA169,76,133,124,169,13,133,125,169,192,133,126,96,141,255,3,104,141,252,
3
1100 DATA104,141,253,3,72,173,252,3,72,201,217,208,7,173,253,3,201,183,240,11,17
3
1110 DATA255,3,201,58,176,3,76,128,,96,104,169,220,72,186,142,254,3,165,113,141
1120 DATA248,3,165,114,141,249,3,162,79,189,,2,157,167,3,202,16,247,56,165,36,22
9
1130 DATA122,168,162,23,201,81,176,73,169,,141,13,3,153,,2,136,177,122,153,,2,13
6
1140 DATA16,248,169,,133,122,169,2,133,123,32,121,165,173,,3,141,250,3,173,1,3,1
41
1150 DATA251,3,169,150,141,,3,169,192,141,1,3,32,115,,32,138,173,162,,173,250,3
1160 DATA141,,3,173,251,3,141,1,3,173,13,3,138,141,13,3,240,6,169,,133,97,133,10
2
1170 DATA173,248,3,133,113,173,249,3,133,114,162,79,189,167,3,157,,2,202,16,247
1180 DATA169,,133,13,174,254,3,154,96,
READY.
```

Kísérletek file-nevek elrejtésére

Ezzel a dologgal is úgy vagyunk, mint a legtöbb védelem trükkel. Ha azokat egyszer nyilvánosságra hozták, akkor rögtön el is vesztik aktualitásukat, hisz így mindenki megismeri őket. Az ilyesfajta trükköket szinte minden számítógépes újságban megtalálhatjuk. Ugyanakkor senki nem említi meg azt, hogy létezik egy egész halom olyan lehetőség, hogy egy file nevének " "-t adjunk. A mai napig hetvennégy Directory megváltoztatási lehetőséget próbáltunk ki a tartalomjegyzék optikai megváltoztatása nélkül, és több mint százat optikai megváltoztatással, így annak a valószínűsége, hogy a helyes megoldást találja ki valaki máris nagyságrenddel lesz kisebb. Egy kis kitartással bizonyára még több lehetőséget is találhatunk, hogyan adjunk csupán két idézőjelből álló nevet egy programnak. De térjünk vissza az említett változatokhoz:

(A kisebb-nagyobb jelek között a megnyomandó billentyű nevét vagy feliratát találjuk.)

1. Az első 15: SAVE CHR$(34)+"1–15 szóköz",8
2. A másodok 15: SAVE "⟨SHIFT+szóköz⟩ és 1–15 szóköz",8
3. Újabb 10 darab:

a) SAVE CHR$(34)+"⟨szóköz⟩,8
b) +"⟨SHIFT+szóköz⟩,8
c) +"⟨CRSR–balra⟩",8
d) +"⟨CRSR–jobbra⟩",8
e) +⟨SHIFT+szóköz⟩"⟨CRSR–balra⟩",8
f) +⟨SHIFT+szóköz⟩"⟨CRSR–jobbra⟩",8
g) +"⟨szóköz⟩⟨SHIFT+szóköz⟩⟨szóköz⟩",8
h) SAVE "⟨SHIFT+szóköz⟩⟨CRSR–balra⟩",8
i) "⟨SHIFT+szóköz⟩⟨CRSR–jobbra⟩",8
j) "⟨szóköz⟩⟨SHIFT+szóköz⟩⟨szóköz⟩",8

4. A legismertebbek:

a) SAVE CHR$(34),8
b) SAVE "⟨SHIFT+szóköz⟩",8

5. A szupernevek (törölhetetlenek!):

a) SAVE CHR$(34)+CHR$(58)+CHR$(34),8
b) SAVE "⟨SHIFT+szóköz⟩"+CHR$(58)+"⟨SHIFT+szóköz⟩",8

6. A három azonos:

a) SAVE CHR$(34)+CHR$(44)+CHR$(34),8
b) SAVE CHR$(34)+CHR$(44)+"⟨SHIFT+szóköz⟩,8
c) lásd az 5a-t!

7. Még 13 fajta:

SAVE"⟨SHIFT+szóköz⟩"+CHR$(58)+"⟨SHIFT+szóköz⟩ és 1–13 szóköz",8

8. Ez a három is van:

a) SAVE "⟨SHIFT+szóköz⟩"+CHR$(44)+"⟨SHIFT+szóköz⟩",8
b) SAVE CHR$(34)+CHR$(58)+CHR$(34)+CHR$(44)+CHR$(34),8
c) mint b) de a középső CHR$(34) helyett ⟨SHIFT+szóköz⟩

9. A legutolsó 14:

SAVE CHR$(34)+"⟨SHIFT+szóköz⟩ és 1–14 szóköz",8

Megjegyzések az elmentési fajtákhoz:

6. pont: Ha ezek közül a nevek közül egyet használtunk, akkor a többit nem vehetjük igénybe! A betöltésnél a CHR$(44) helyett használhatjuk a CHR$(58)-at és fordítva! A tárolásnál ugyanez igaz.

Az optikai befolyásoláshoz a ⟨DEL⟩ billentyűt használhatjuk. De a ⟨SHIFT+szóköz⟩ a funkciós billentyűkkel együtt szintén megteszi a magáét.

Sok örömet kívánok a kipróbáláshoz és a további kereséshez!

Hol a programom?

Ha a programok tárolására a Datasette-et használjuk, akkor bizonyára előfordult már, hogy nem jegyeztük meg, hol is áll a kazettán a kívánt program. Ha a kazettát egy normál magnóba helyezzük, akkor a sípoló hangokból meg lehet állapítani a programok kezdetét és végét. A mellékelt háromsoros program használatával ugyanezt a C64-gyel is megcsinálhatjuk.

```
10 S=53218:FOR L=S TO S+29:READ P:POKE L,P:NEXT:SYS S
20 DATA 120,169,7,133,1,169,0,141,17,208,162,0,173,3,220,41,16
30 DATA 240,5,238,32,208,162,15,142,24,212,24,144,236
```

Pötyögjük be a programot, majd indítsuk el RUN-nal. Helyezzünk be egy programokkal teli kazettát a Datasette-be, és nyomjuk meg a PLAY billentyűt. Ekkor hosszú füttyhang jelzi nekünk, ha egy program kezdődik. Idővel biztosan eléggé vájtfülűek leszünk a programok lehallgatásához!

Előfordulhat az is, hogy a gépben van egy program, amit mindenképpen tárolnunk kell, de nem tudjuk, hol ért véget egy másik program a kazettán. Meddig tekerjük hát a szalagot?

A megoldás egyszerű. Adjuk ki a VERIFY utasítást. A magnót elindítva pontosan a kiválasztott program végére jutunk. Természetesen VERIFY ERROR hibajelzést kapunk, de ez mindegy, mi máris SAVE-elhetünk...

Reset elleni védelem

Rendes körülmények között egy utólag beépített reset kapcsolóval minden programot meg lehet szakítani, és a C64-est a bekapcsolás utáni állapotba vihetjük. A gépünk azonban rendelkezik egy olyan lehetőséggel, hogy egy, a bővítő portba csatlakoztatott programmodult azonnal a bekapcsolás után elindítsunk. Ehhez csupán arra van szükség, hogy a tárolóban egy bizonyos helyen (a $8004-es hexadecimális címtől kezdve) a cbm80 jelsorozat álljon. Ezt a tárolóterületet mi is fölülírhatjuk, így a reset kapcsoló hatását kikerüljük. A „RESETVEDELEM" program egy ilyen modult szimulál.

```
2 FOR T=5300 TO 53065:READ A:D=D+A:POKE T,A:NEXT
3 IF D<>7352 THEN PRINT ,,DATAHIBA"
4 SYS 53000
5 DATA 162,5,189,15,253,157,3,128,202,208,247,169,41,141
6 DATA 0,128,141,2,128,169,207,141,1,128,141,3,128,169
7 DATA 52,141,20,3,96,32,163,253,32,83,228,32,91,255
8 DATA 88,169,82,141,119,2,169,213,141,120,2,169,13,141
9 DATA 121,2,169,3,133,198,108,2,160,0
10 itt kezdődhet a saját programunk
```

Ha ezeket a sorokat beépítjük a saját programjainkba, majd azt egyszer fölhívjuk, akkor nem lehet többé resetet kiváltani. A C64-es ugyanis ebben az esetben egy RUN parancsot fog végrehajtani.

GOTO X és GOSUB Y

A BASIC programozó tudja, hogy a GOTO és a GOSUB utasítás paramétere csak konstans lehet. A GOTO X-szerű, változót tartalmazó megoldás nem működik. Egy apró trükkel azonban ez is megvalósítható.

Először a csak olvasható ROM tartományt az írható és olvasható RAM-ba kell vinnünk:

```
10 FOR K = 40960 TO 49151: POKE K,PEEK(K):NEXT
```

Első pillantásra a rutin értelmetlennek tűnik. Egy értéket olvasunk ki egy cellából, majd azt ugyanoda visszaírjuk. Gyakorlatilag azonban nem történik más, minthogy kihasználjuk a C64 egy sajátosságát. Ha ugyanis a (fölülírhatatlan) ROM tartományra írással (pl. POKE) hivatkozunk, az operációs rendszer a műveletet az azonos helyen található RAM-ba irányítja. A fönti rutin a ROM-ból olvassa ki az értéket, amelyet azután a RAM-ba írunk. Ezután még nem lesz probléma az operációs rendszer fölülírása, módosítása.

A GOTO és GOSUB utasítások módosításához az új rutint az operációs rendszer egy szabad területére kell írnunk. Ilyen terület pl. a 48999—49004 tartomány:

```
20 FOR K = 48999 TO 49004:READ A:POKE K,A :NEXT
30 DATA 32, 138, 173, 76, 247, 183
```

A 30-as sor DATA értékei egy gépi kódú programot rejtenek. Ennek részletei itt nem érdekesek. Viszont két további POKE utasítás szükséges, hogy a régi rutin helyett az újat hajtsuk végre:

```
40 POKE 43169, 103:POKE 43170, 191
```

Próbáljuk ki most a GOTO és GOSUB utasításokat. Első meglepetésünkre egyelőre semmi nem változott! Ennek oka az, hogy „elfelejtettük'' közölni a C64-gyel, hogy mostantól az operációs rendszer programot ne a ROM-ban, hanem a RAM-ban dolgozza föl. Egy egyszerű POKE-kal ez is megoldható:

```
50 POKE 1, 54
```

Ettől a pillanattól kezdve a GOTO X és a GOSUB Y a rendelkezésünkre áll. Ne feledjük el azonban azt, hogy az ON X GOTO és az ON Y GOSUB utasításoknál ezt a változást nem hajtottuk végre!

Képernyővédő BASIC sorok

A mellékelt program olyan rutin, amely alprogramként saját BASIC programjainkban fűzhető. A cél, hogy a GET lekérdezések alatt a képernyőt egy bizonyos idő elteltével lekapcsoljuk. Ezzel megakadályozható az, hogy a képernyőn hosszú ideig „mereven” álló karakterek beégjenek oda.

A legtöbb ilyen rutint gépi kódban írják, de ott gondoskodni kell a tárolóban való elhelyezésről. A mi megoldásunk tisztán BASIC (néhány POKE-tól eltekintve), így ez bármely programban tetszés szerint fölhasználható anélkül, hogy nekünk kéne az elhelyezésével és az összeférhetőségével törődni.

A használathoz a főprogramban a Z változóba kell tenni azt az értéket (másodpercben), amelyet szükségesnek tartunk. Ha ez az idő úgy telik el, hogy nem nyomtunk meg egy gombot sem, a képernyőt lekapcsoljuk. A használat másik feltétele, hogy a főprogram kérdéses GET lekérdezése előtt a T változót a TI/60 értékre állítsuk be. A GET után kell az alprogramot fölhívni. Ha a képernyőt gombnyomás híján a megadott idő eltelte után lekapcsoltuk, elegendő egy billentyűt működtetni, és a kijelzés ismét „eleven” lesz.

A mellékelt listában a program a 25—60. sorokban áll, az 1—7. sorok csak a bemutatás kedvéért vannak.

```
1 REM *** KEPERNYO VEDELEM ***
2 Z=15:T=TI/60:PRINT "♥"
3 GET A$:GOSUB 10:REM BOLLENTYU LEKERDEZES ES ALPROGRAM HIVAS
4 PRINT A$;:GOTO 3:REM A LEKERDEZES FELDOLGOZASA
6 REM ********************************
7 :
10 REM KEPERNYO KOKAPCSOLO SZUBRUTIN
20 :
25 IF A$<>"" THEN T=TI/60:RETURN
30 IF T+Z>TI/60 THEN RETURN
40 AL=PEEK(53280):POKE 53265,PEEK(53265) AND 239:POKE53280,0
50 GET AC$:IF AC$="" THEN 50
60 POKE 53265,PEEK(53265) OR 16:POKE53280,AL:T=TI/60:RETURN
```

Bitek be- és kikapcsolása

A BITSETTER program egy parancsbővítés a C64-hez. A Commodore BASIC a program használata esetén a BIT paranccsal lesz gazdagabb, amivel egyes biteket sokkal könnyebben lehet majd be- és kikapcsolni.

Például: A blokkgrafika/multicolor módba eddig az alábbiak szerint kellett átkapcsolni:

POKE 53248+22,PEEK(53248+22) OR 16

A 22. grafikus regiszterben bekapcsoltuk a 4. bitet. A BIT paranccsal ez sokkal egyszerűbben megy:

BIT,53248+22,4

Ezzel pedig ki tudjuk küszöbölni a logikai műveleteknél jelentkező problémákat, ami különösen a kezdők számára jelent problémát.

A programot a LOAD „BITSETTER.OBJ”,8,1-gyel kell betölteni. Az indítása pedig a NEW parancs beadása után a SYS 49152-vel történhet. Az új utasítást úgy parancs-, mint programmódban használhatjuk.

SZINTAXIS: BIT,tárolócím,bit

Példa: Be akarjuk kapcsolni az 5. sprite-ot. Ez eddig így történt:

POKE 53248,21,PEEK(53248+21) OR 32

A megfelelő BIT parancs pedig:

BIT,53248+21,5

Ha az 5. bit be van kapcsolva, akkor az kioltódik, ha ki van kapcsolva, akkor értéke egy lesz. Ezentúl az ilyen műveletekhez csupán egy parancsra lesz szükségünk. A mindenkor meghívott bitet tehát átkapcsoljuk. Ha tehát a sprite-ot ismét ki akarjuk kapcsolni, újból ki kell adni az előző utasítást.

A BITSETTER program.
A beviteléhez használjunk egy monitorprogramot.

```
c000:  a9  0b  8d  08  03  a9  c0  8d
c008:  09  03  60  20  73  00  c9  42
c010:  f0  06  20  79  00  4c  e7  a7
c018:  20  1e  c0  4c  ae  a7  20  73
c020:  00  c9  49  d0  2f  20  73  00
c028:  c9  54  d0  28  20  73  00  20
c030:  fd  ae  20  eb  b7  bd  57  c0
c038:  8d  50  c3  a6  14  8e  4c  c0
c040:  8e  52  c0  a6  15  8e  4d  c0
c048:  8e  53  c0  ad  00  c0  4d  50
c050:  c3  8d  00  c0  a9  20  60  01
c058:  02  04  08  10  20  40  80  03
```

Tippek a Hardmaker programhoz

A Hardmaker programot jól kell hogy ismerjék mindazok, akik grafikákkal foglalkoznak. Erről a Commodore Újság hasábjain is olvashattunk már. Bár a valóban kitűnő Hardmaker problémamentesen együttműködik az Epson kompatibilis nyomtatókkal, mégis lehet néhány dolgot javítani:

1. A soremelés 7/72 collra van beállítva. Viszont hogy a sorok ne fedjék egymást, 8/72 collos továbbítás szükséges.
2. A képernyőt csak 66%-os szélességben nyomtatjuk ki.
3. A 2. pont miatt a grafika vízszintesen torzít.
4. A világosszürke és a sötétszürke tónusokat a multicolor módban fölcserélve adjuk ki.

A változtatásokhoz csupán egy gépi kódú monitor szükséges. Vegyük elő azt, majd töltsük be a Hardmakert.

1. A soremelés beállítása 8/72 collra (ESC ,,A" 08):

```
:150D A9 1B 20 21 15 A9 40 20
:1515 21 15 EA
:1540 03 C0 4C 1B 08 41 1B
,141A LDY #$02
,1464 LDA #$02
```

2. A szürketónusok helyes kivitele multicolor üzemmódban:

```
,149A LDA 1560,Y
,14A4 LDA 1554,Y
```

3. Változtatások a HiRes-kivitelnél (képernyő formátum, soremelés):

```
:16F0 41 08 FF FF 1B 2A 06 FF
:16F8 FF 1B 40
```

A változtatások után a Hardmakert S,,új név",0801,1720 utasítással ismét le kell tárolni. Természetesen lehetséges, hogy nekünk egy már javított változat van a birtokunkban, ebben az esetben nincs mit javítanunk.

Törtek egyszerűsítése

Gyors prímszámokat kiszámító programokkal ma már Dunát lehet rekeszteni. Viszont még senki sem készített eleddig egy gyors törtegyszerűsítő programot. Nos a mellékelt lista pont egy ilyen problémamegoldást ad.

Először meghatározzuk a számláló és a nevező legnagyobb közös osztóját, egy euklideszi algoritmus szerint (40. sor), a többi azután már gyerekjáték.

```
10 INPUT"számláló";SZ
20 INPUT"nevező";NE:
30 A=SZ:B=NE
40 R=A-INT(A/B)*B:A=B:B=R:IF R)0 THEN 40
50 PRINT:PRINT"az egyszerűsített eredmény:":PRINT
60 PRINT"a számláló:"SZ/A:PRINT
70 PRINT"nevező: "N/A
```

Működik a nyomtató?

Gyakorta adódik olyan helyzet, hogy egy programból meg kell állapítani, hogy a nyomtató be van-e kapcsolva. A problémát mindenféle gépi kódú program vagy POKE sorok nélkül is meg lehet oldani:

```
10 OPEN1,4,1:CLOSE1:IF ST=–128 THEN PRINT
,,NYOMTATÓ!": GOTO 10
```

Kikapcsolt printer esetén ez a sor végeláthatatlan sorokban írja a ,,NYOMTATÓ" feliratot a képernyőre, de programozhatunk várakozóhurkot, tetszőleges helyre is ugorhatunk, vagy mondjuk egy alprogramba léphetünk.

Több blokk a sprite-ok számára

Általában a sprite adatait a kazettapufferban szoktuk elhelyezni, vagypedig a BASIC program kezdetét jelző mutató az adott program betöltése előtt magasabbra toljuk. A megadott program a BASIC kezdetet úgy tolja el, hogy plusz öt blokk szabadul fel.

```
10 POKE 44,10:POKE 10*256,0
20 PRINT"{3 kurzor le} LOAD"CHR$(34)"A PROGRAMUNK
   NEVE"CHR$(34)",8
30 PRINT"{4 kurzor föl}"
40 POKE631,13:POKE632,82:POKE633,213:POKE634,13:POKE 198,4
50 NEW
```

A hét napjának kiszámítása

A mellékelt kis program egy rövid öröknaptár, a hét napjainak kiszámításához, amely egészen a 3333. évig jó.

Először be kell adni a napot a hónapot és az évet, (01.01.0001), majd meg kell nyomni a ⟨RETURN⟩ billentyűt. Röviddel ezután megjelenik a képernyőn az adott dátumhoz tartozó nap.

```
100 PRINT CHR$(147);:FOR I=0 TO 6:READ W$(I) :NEXT
110 INPUT"NN.HH.EEEE";A$
130 T=VAL (MID$(A$,1,2)):M=VAL (MID$(A$,4,2)):J=VAL
    (MID$(A$,7,4))
160 X=T+M*31–31–INT (M*.43) +(M>2)*2+(J–1)*365+
    INT((J+(M<3))/4)
170 X=X+(((J+4)/400=INT((J+4)/400))AND (M)2))–INT ((J+104)/100)
    INT((J+104)/400)
180 Y=X–INT(X/7)*7
190 PRINT:PRINTW$(Y):PRINT:PRINT
200 GOTO110
300 DATA VASARNAP,HETFO,KEDD,SZERDA, CSUTORTOK, PENTEK,
    SZOMBAT
```

Tarka REM

10 DATA 169,11,141,8,3,169,192,141,9,3,96
20 DATA 32,115,0,201,143,240,6,32,121,0,76,231,167,238,32,208,76,18,192
30 FOR A=49152 TO 49181:READ X:POKE A,X:NEXT:SYS49152

Az itt megadott kis program hatása az, hogy ha a programban egy REM sort szerepel, akkor a keret színe eggyel magasabb lesz. Ezzel például meg lehet állapítani, hogy a programunk fut-e, vagy lemerevedett. Ezen kívül más célokra is föl lehet ezt a lehetőséget használni, amit a mellékelt demoprogramok hivatottak szemléltetni:

```
Demo 1: 10 REM
        20 REM
        30 REM
        40 REM
        50 REM
        60 REM
        70 REM
        80 REM
        90 REM
        100 REM
        110 REM
        120 REM
        130 GOTO 10

Demo 2: 10 X=X+1000
        20 FOR A=1TOX
        30 REM
        40 REM
        50 REM
        60 REM
        70 REM
        80 IFX=1000
        THEN10

Demo 3: 10 REM
        20 REM
        30 GETA$:IF
        A$=""THEN10

Demo 4: 10 REM
        20 REM
        30 REM
        40 REM
        50 REM
        60 GETA$:IF
        A$=""THEN10
```

Hexadecimális/decimális átváltás

Egy hexadecimális szám decimálissá történő átváltásához általában több programsorra van szükség. Az alábbi egysoros ugyanezt tetszőleges hexadecimális számra (H$) megteszi.
10 D=0:A=LEN(H$):FORI=1TOA:B=ASC(MID$(H$,I):D=D+(B—48+7*(B>64))*16(A—I):NEXT

Ha a parancsokat rövidítve adjuk be, akkor sorszámnak nagy számokat is vehetünk, és egy RETURN parancsot is beilleszthetünk. A programban először a D decimális számot nullázzuk. Ezt követően egymás után minden egyes számjegy ASCII értékét a B-be visszük. Ebből 48-at levonunk, illetve ha az érték nagyobb mint 64, akkor még hetet, és kiszámítjuk a karakter értékét. Ezt azután megszorozzuk a hozzátartozó 16 hatvánnyal, majd hozzáadjuk a D-hez. A következő listában ennek az ellenkezője történik. Ezt a sort egy programsorba kell bevinnünk:

```
1 H$="":FORI=DTO-1:A=INT(I/16)*16-IAND15:H$=CHR$(48+
A-7*(A>9))+H$:I=I/16-1:NEXT
```

ezért választottuk sorszámnak az 1-et. Ahogy az első egysorosban is, a D tartalmazza a decimális, a H$ a hexadecimális számot.

Bináris/Decimális átszámítás

1. Decimális szám átszámítása binárissá:

```
5 B$="":N=16: FOR T=1 TO N: A=(Z<2↑(N-T))+1:
B$=B$+CHR$(A+48): Z=Z-A*2↑(N-T): NEXT
```

N = a bináris alakú szám helyeinek száma
B$ = a bináris szám sztringformátumban
Z = a decimális szám (a rutin fölhasználása előtt kell rögzíteni)

2. Bináris szám átalakítása decimálissá:

```
5 Z=0:A=LEN(B$)-1: FOR T=0 TO A:Z=Z+VAL (MID$
(B$,A+1-T,1))*2↑T:NEXT
```

B$ = a bináris szám sztringformátumban (előbb rögzíteni kell)
Z = a decimális szám.

Két program összefűzése

Az alábbi rutinnal lehetőségünk lesz arra, hogy két programot egymással összekapcsoljunk. Csupán arra van szükség, hogy a második program első sora nagyobb legyen mint az első program utolsó sora. Járjunk el az alábbiak szerint:

1. Töltsük be az első programot.
2. Adjuk be a PRINT PEEK(43),PEEK(44) utasítást. Ebben a két tárolóban található a BASIC program kezdőcíme. Jegyezzük föl mindkét kijelzett számot. Ezután adjuk be a
3. POKE43,(PEEK(45)+256*PEEK(46)—2)AND255 és a
4. POKE44,(PEEK(45)+256*PEEK(46)—2)/256 parancsot. (A 45-ös és a 46-os tárolóhely tartalmazza a BASIC program végének címét.)
5. Töltsük be a második programot.
6. POKE 43, az első földjegyzett szám.
7. POKE 44, a második földjegyzett szám.

Ha minden lépést rendben végrehajtottunk, akkor a LIST utasítás után csupán egyetlen egy program lesz a tárolóban.

Kizáró vagy BASIC-ben

A C64 BASIC-je csak az AND, OR és NOT logikai függvényeket kínálja. Ha valamilyen egyéb Boolean elemre van szükségünk, magunknak kell megvalósítanunk azt. Az egyik legtöbbször igényelt elem a kizáró vagy, amelyet a legkönnyebben az alábbi képlettel valósíthatunk meg:

```
A = (X OR Y) - (X AND Y)
```

Az AUTO parancs

A C128-asnál az AUTO 10 utasítás kiadása után minden beadott sort követően automatikusan megkapjuk a következő, tízzel nagyobbat. Ez jó segítség, ha mondjuk programot írunk vagy adunk be papírról. A C64-es esetében eddig ez nem volt lehetséges. Két BASIC sor segítségével azonban mégis menni fog:

```
1 PRINT CHR$(147)A;:POKE 19,1:INPUT A$:POKE 19,0:PRINT
2 PRINT"A="A+10":GOTO1":POKE 198,3:POKE 631,19:POKE 632,13:
POKE 633,13:END
```

A programot az A=10:GOTO1-gyel kell indítani. Ezt követően minden sor beadása után kiadjuk a következő sorszámot, ami itt tízzel lesz magasabb mint az előző. Ha nem 10-et akarunk a 2. sorban írjuk át az "A+10" részt, pl. 5-nél "A+5"-re. A rutint a ⟨RUN/STOP⟩ ⟨RESTORE⟩ billentyűkkel lehet elhagyni.

A mini szövegföldolgozás

Ha éppen nincsen kéznél egy szövegföldolgozó program, de okvetlenül szükség lenne egy szöveg kinyomtatására, akkor azt egy apró trükk segítségével megtehetjük. Legelőször meg kell írni a kívánt szöveget úgy, hogy minden sor elé egy BASIC sorszámot írunk. Például:
10 A Commodore Újság egy csúcs lap!
Ezután jöhet a tulajdonképpeni trükk:
1. Kapcsoljuk be a nyomtatót.
2. Adjuk be a POKE 22,35:OPEN4,4:CMD4:LIST utasítást.
3. Adjuk be a PRINT#4:CLOSE4:XXX utasítást.

Az „XXX" egy SYNTAX ERROR hibajelzést okoz, ami a számítógépet visszaállítja az eredeti állapotba. A szövegünk azonban már kész van. Az egyetlen hátrány: Egy szöveges sor soha nem kezdődhet számmal!

Kézbentartott joystick

A kezdő programozók, akik joystick lekérdezést programoznak, a legtöbbször „eltévednek" a PEEK lekérdezések IF-THEN dzsungelében. Mindez kényelmetlen, zavaró és még hosszú is. Bezzeg a C128! Ott elég egy parancsot kiadni, máris megvan az eredmény. Van azonban számunkra is megoldás. Nem olyan rövid mint a 128-as, de ugyanolyan jól használható. Íme:

```
DEF FN JOY(X)=INT((LOG255.5-(PEEK(56322-X)OR224)))/
        LOG(2)+2)
```

Ettől kezdve **PRINT FN JOY(X)** utasítással bármikor lekérdezhetjük az X (=1,2) portba dugott joystick állását. A kapott eredménynél 1=alaphelyzet, 2=fönt, 3=lent, 4=balra, 5=jobbra és 6=tűzgomb. Sajnos kombinált lekérdezés (pl. tűzgomb és föl egyszerre) ezzel a módszerrel nem lehetséges.

Egy **ON FNJOY(X) GOTO**... sor segítségével máris alprogramokra léphetünk, kijelzést módosíthatunk vagy sprite-ot vezérelhetünk kedvünk szerint.

Képletek BASIC-ben

Ha az általunk írt programban kívülről képleteket kell bevinni, az legtöbbször igen bonyolult lesz. Leggyakrabban egy komplikált rutint használunk erre a célra, hogy a képletszámítás megtörténhessen. Sokkal könnyebben megy azonban a dolog, ha erre a célra a képernyőt használjuk. Ekkor csak a képlet szintaktikai ellenőrzése marad. Az ajánlott BASIC sor:

```
1 PRINT"{lig.blue,clr}"A$"="A:INPUTA$.:PRINT"{home,blue} A=
"A$":GOTO 1":POKE 631,19:POKE 632,13:POKE 198,2
```

Figyelem:

Ne keressük se a kapcos zárójelet, se a „lig.blu" stb. karaktereket a billentyűzeten. Ezzel csak a bevitelt kívánjuk egyszerűsíteni, az inverz és kitalálhatatlan BASIC kódok helyett. Tehát az idézőjelben a világoskék szín ⟨C=7⟩, a clr ⟨SHIFT HOME⟩ stb. kódokat adjuk be!

Az első indításnál a képernyőn az "=0" jelenik meg képletként (a pont az INPUT A$ után!). Adjuk be most a kiszámítandó képletet. A következő lépésben a képernyőre írjuk az „A=" képlet kifejezést, illetve egy ugrási parancsot. A billentyűzet pufferbe két karaktert írunk (home és return). Mihelyst a programunknak vége, ezt a két jelet kiértékeljük. Ezzel a trükkel számítjuk ki tulajdonképpen a kifejezés értékét.

A képletet ebben a rövid formában nem ellenőrizzük, így hiba esetén nincs jelzés. Mivel ilyenkor nem váltunk vissza a kék (írás) színről világoskékre, azt előbb nekünk kell megtenni. Ha nem sötétkék lenne a háttérszín, módosítani kell a második PRINT utasítás színértékét. Ellenkező esetben az egész „trükk" olvasható lesz a képernyőn.

Mi az a compiler?

Igen gyakran kérdezik tőlünk az olvasóink, mi is az a compiler tulajdonképpen, és mit jelent a „compilálni" kifejezés.

Egy BASIC compiler egy olyan program, amely a BASIC programokat gépi kódú programmá fordítja le. Ha tehát egy programot „compilálunk", akkor az sokkal gyorsabb lesz. A lefordított programot „compilátumnak" is szokták nevezni. Ezt a normál módon lehet betölteni és RUN-nal indítani, viszont LISTáztatni már nem. Ugyanis a gépi kódú programokat nem lehet úgy kilistázni, mint a BASIC-ban írtakat. Ennek oka viszont nem egy LISTavédelem, hanem a gépi kódú programok sajátossága. Ha egy gépi kódban írt programot analizálni akarunk, akkor egy úgynevezett „gépi kódú monitorra" van szükség, meg egy kis programozói gyakorlatra.

Compilereket vásárolhatunk is. Ilyenek például az „Austrocomp", az „Austrospeed", a „BASIC 64" stb. programok is.

Hibás monitor?

Ez a BASIC egysoros ide-oda rázogatja a képernyőt:

```
0 FOR A=0 TO 15: POKE 53270,A:NEXT:GOTO 0
```

PÖTYÖGŐ SZOLGÁLAT

251	C64	90/12	SPRITE PEEPER	70.-
252	C64	90/12	SPRITE	50.-
253	+4	90/12	Malom	140.-
254	C64	91/ 1	Vizszintes scroll	50.-
255	C64	91/ 1	BASIC betöltö készitö	50.-
256	C64	91/ 1	VERIFY MASTER	70.-
257	C64	91/ 1	Hibamentes replace	100.-
258	+4	91/ 1	Auto Input	100.-
259	C64	91/ 1	Törtek	50.-
260	C64	91/ 1	Ablakozás könnyen	70.-
261	+4	91/ 2	80-as képernyö	60.-
262	C64	91/ 2	Fizikateszt	40.-
263	C64	91/ 2	40 Sprite	100.-
264	C64	91/ 2	Raster - Master	60.-
265	C64	91/ 3	Rulett	80.-
266	+4	91/ 3	Tape-disc copy	90.-
267	C64	91/ 3	BASIC rearranger	40.-
268	C64	91/ 3	BASIC decompactor	40.-
269	C64	91/ 3	BASIC merger	40.-
270	C64	91/ 3	SEIKOSHA	60.-
271	C64	91/ 3	Cartridge szimuláció	100.-
272	+4	91/ 3	Sally kulcsszókeresö	40.-
273	+4	91/ 3	Merge C+4	60.-
274	C128	91/ 4	1.5 MHz-es C64	80.-
275	C64	91/ 4	Morze - H	60.-
276	C64	91/ 4	Gprint	40.-
277	C64	91/ 4	ASCII - CHR$	40.-
278	C64	91/ 5	Rendezö	60.-
279	C64	91/ 5	Keretbeiró	50.-
280	C64 +4	91/ 5	NLQ Print	50.-
281	C64	91/ 5	FLD	40.-
282	C64	91/ 5	Raster Split	40.-
283	+4	91/ 6	Programnyilvántartó	60.-
284	C64	91/ 6	Regiszter	50.-
285	C64	91/ 6	Digitalizált zene	40.-
286	C64	91/ 6	Ékezetes MPS 802	60.-
287	C64	91/ 7	*Ékezetes iratkészitö	80.-
288	+4	91/ 7	Büvös téglalap	80.-
289	C64	91/ 7	Dominó - H	80.-
290	C64	91/ 7	Raktáros	60.-
291	+4	91/ 7	Tologató	50.-
292	C64	91/ 7	Aren 2000	40.-
293	C64	91/ 7	Bad Lamps	60.-
294	C64	91/ 7	Operation Patriot	40.-
295	+4	91/ 7	Mozaik - B	70.-
296	+4	91/ 7	Logi Kód	50.-
297	C64	92/ 1	Databevivö	40.-
298	+4	91/ 7	Bomber	40.-
299	+4	91/ 7	Dominó - K	40.-
300	+4	91/ 7	The Wall	30.-
301	C64	91/ 7	Mozaik - K	40.-
302	+4	91/ 7	Dominó - R	50.-
303	+4	91/ 7	*Borgok Kincse	150.-
304	+4	91/ 7	Asteroids	50.-
305	C64	91/ 9	Ékezetes irat forrás	30.-
306	C64	91/ 9	Yugo2	40.-
307	C64	91/ 9	Adventure - Sz	100.-
308	C64	91/ 9	Morze Oktató B.	100.-
309	+4	91/ 9	Memory Game Plus	80.-
310	C64	91/ 9	*Capitaly	100.-
311	C64	91/ 9	*Gengszterek	50.-
312	C64	91/ 9	*Küldetés 2000	80.-
313	C64	91/ 9	Itt a $ hol a $	40.-
314	C64	91/ 9	Hewing	40.-
315	+4	91/10	Mikrobi	40.-
316	+4	91/10	Nest of Fleas	150.-
317	+4	91/10	Adventure - F	80.-
318	+4	91/10	A világ szeme	120.-
319	C64	91/10	Gondolkozó	80.-
320	C64	91/10	Memori - B	100.-
321	+4	91/10	Matekastély	80.-
322	C64	91/11	Lottó	60.-
323	C64	91/11	Totó	60.-
324	C64	91/11	Német gyakorló	50.-
325	+4	91/11	Beszélö szótár	100.-
326	+4	91/11	File examiner	50.-
327	C64	91/11	Sprite szerkesztö	40.-
328	C64	91/11	Egyszinü Sprite	40.-
329	C64	91/12	Rulett javitás	40.-
330	C64	91/12	Tözsde	100.-
331	C64	91/12	IQ teszt	120.-
332	C64	92/ 1	Kalandlap	60.-
333	C64	92/ 1	Levélíró	60.-
334	C64	92/ 1	Képernyömaszkoló	100.-
335	C64	92/ 1	Sprite Kezelö	80.-
336	C64	92/ 1	SI - OPART	30.-
337	C64	92/ 2	Patience Quartet	100.-
338	C64	92/ 2	Monopoly	100.-
339	C64	92/ 2	Akasztófa	60.-
340	C64	92/ 2	Karaktertervezö	50.-
341	C64	92/ 3	String Array Manager	80.-
342	C64	92/ 3	Sprite Basic	40.-
343	C64	92/ 3	Screen Help	40.-
344	C64	92/ 3	Gyors validate	50.-
345	C64	92/ 3	*Az alvilág ura	100.-

A *-gal jelölt programok csak lemezen rendelhetök !

Kérem, hogy a **Commodore Újság** legközelebbi számában jelenjen meg a következő szövegű apróhirdetés:

(minden kockába egy betűt írjon)

Tagsági szám: ....

Feladandó az újság címére:
Commodore Újság
Budapest, 1388. Pf. 86.

Alulírott megrendelem a következő programokat a PÖTYÖGŐSZOLGÁLAT-tól:

PROGRAM SORSZÁMA **ÁRA**

1. ,– Ft
2. ,– Ft
3. ,– Ft
4. ,– Ft
5. ,– Ft
6. ,– Ft
7. ,– Ft
8. ,– Ft
9. ,– Ft

Összesen: db ,– Ft

A programokat a SZOLGÁLAT által biztosított
lemezre (99 Ft/db)
kazettára (40 Ft/db)
az általam küldött adathordozóra kérem.
(Kérjük a megfelelő szöveg aláhúzását!)
Postaköltség (35 Ft)
A fizetés módja: személyesen — csekken — utánvéttel
(Kérjük a megfelelő szöveg aláhúzását!)
ÖSSZESEN:

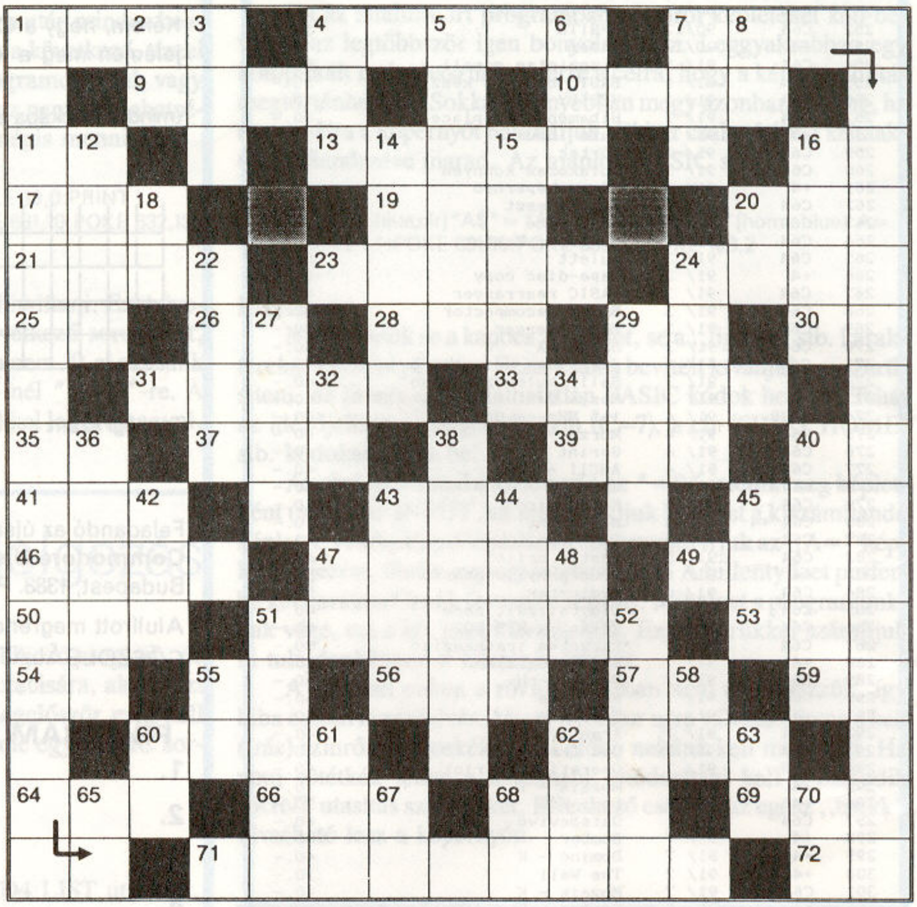
A megrendelő neve: ....

Címe: ....

KERESZTREJTVÉNY

EURIPIDÉSZ (i.e. 480—406): az egyik legnagyobb athéni drámaíró egyik elmés mondása van elrejtve a vízszintes 71., függőleges 5., 1., vízszintes 7. sorokban.

A meghatározásokat nem választottuk külön, hanem a számozás sorrendjében folyamatosan adjuk. Ha egy számtól mindkét irányba indul ki a megfejtés, akkor előbb a vízszintes, majd egy gondolatjellel elválasztva a függőleges meghatározása szerepel.



MEGHATÁROZÁSOK: 2. Kettős betű. 3. Lakoma. 4. Iparos. — Fél szüret. 6. Kémiai elem? 7. Vízinövény. 8. Betűt vet. 9. Ízesíts. 10. Évszak. 11. Másik kettős betű. 12. Francia író volt (Emile). 13. Verőre. 14. Idegen ügetés. 15. Alarm. 16. Azonos a 8. függőlegessel. — Hivatalos papír. 17. Ilyen szó is létezik. 18. Irányszó. 19. Éri keverve. 20. Kor. — Hiánytalan. 21. Válogatott. 22. Sajátod. 23. Szemétdomb ura. 24. Dömöshöz tartozó település. — Felső ellentéte. 25. E nap. 26. Olaszországi és ausztriai gépkocsik jelzései. 27. Női név. 28. Ilyen lábápolás is van. 29. Thaiföldi és luxemburgi gépkocsik jelzései. — Város Komárom megyében. 30. Egyforma betűk. 31. Gólya beszéd. 32. Egon páratlan betűi. 33. Baranya megyei helység. 34. Debreceni állat. 35. Vés. 36. Televízió márka. 37. Héjas gyümölcs. 38. Szerencsét hozó tárgy. 39. Ilyen kórházi kezelés is van. 40. Kötőszó. — Egy bizonyos helyre puskát süt el. 41. Dyn. 42. Elődje. 43. Szilaj. — Becézett női név. 44. Sakkban: királynő, vezér. 45. Férfinév — ...ptív, lobbanékony. 46. Becézett férfinév. 47. Buda egyik része. — Tantál vegyjele. 48. Nobélium vegyjele. 49. Kitüntetés. 50. Éktelen bárkás ember. 51. 11 évig uralkodott e király. — Vonat fajta. 52. Ez a tied. 53. Város Németországban. 54. Magam. 55. Irányszó. — Ugyanaz mint a vízszintes. 56. Becézett leánynév. 57. Mutatószó. 58. Fordított névelő. 59. Őszi határok. 60. Ritka férfinév. — Stroncium vegyjele. 61. Évszak. 62. Nem tisztességesen játszik. — Van ilyen egyenes is. 63. Szolmizációs skála. 64. Ilyen út is létezik. — Gyilkol. 65. Napszak. 67. Személyes névmás. 68. Kémiai elem. — Rénium vegyjele. 69. Nem fölé. 70. Folyadék. 72. Egy másik napszak.

Directory bújócska

Formatáljunk egy lemezt az alábbiak szerint:

```
OPEN 15,8,15,"N:név"+CHR$(0)+CHR$(0)+CRH$(0)+",id"
```

A directory következő betöltésekor kizárólag a nevet kapjuk vissza. Ennek oka az, hogy a LIST rutin három nullát a BASIC program végének értelmez és befejezi a munkát. Mindezek ellenére a tartalomjegyzékünk teljesértékű lesz.

Másik trükk, ha a programjaink kimentésekor a név mögé egy ",S,W"-t rakunk: SAVE "próba,s,w",8

Ebben az esetben a PRÓBA file nem PRG, hanem SEQ-ként kerül kijelzésre, amelyet természetesen a LOAD „PROBA",8 utasítással nem lehet betölteni (64, FILE TYPE MISMATCH ERRROR).

Kizárólag a LOAD "PRÓBA,S,R",8 vezet eredményre. Ha többféle „bújócskát" kombinálunk, hatékonyabb védelmet kapunk.

...és még egy directory

A valóban legegyszerűbb módszer a tartalomjegyzék programvesztés nélküli megtekintésére az alábbi egysoros:

```
POKE 254,PEEK(45):POKE 255,PEEK(46):POKE 44,192:LOAD"$",8
```

Ezt követően kilistázhatjuk a directoryt. Minden RUN, NEW vagy hasonló parancs OUT OF MEMORY ERROR-hoz vezet. Ha tovább akarunk dolgozni, három POKE kell:

```
POKE 45,PEEK(254):POKE 46,PEEK(255):POKE 44,8
```

A dolog lényege az, hogy a tárolóban álló program lényeges mutatóit úgy állítjuk át, hogy azok a meglévő program mögé mutatnak. Természetesen oda bekerülhet a directory. A munkát pedig az értékek helyreállítása után zavar nélkül folytathatjuk.

C-64-re 92-es programok eladók 5000-es választékból K&C COMPUTER.
3630 Putnok, Pf.: 25.
Karszky János 3635 Dubicsány, Rákóczi u. 3.

1 éves AMIGA 500 1 MB bővítő, infrajoy, egér, 80 lemez, TV modulátor, szakkönyvek 51 000 Ft-ért eladó.
Érdeklődni:
Izsó Róbert, 1134 Budapest, Kassák L. u. 76. IX/83.

Figyelem! Eladó egy C-64 II. 1541 II. floppy-drive, 60 db lemez (tele a legújabb programokkal), Action Cartridge, magnó, kazetták, irodalom! Irányár: 40 000 Ft Nem bánja meg !
Karczag Zoltán, 30014 Hort, Kossuth u. 107.

Amigához 5.25-ös drive 12 000 Ft. Ezenkívül sikerprogramok óriási választéka!
Barabás Zsolt, 2230 Gyömrő, Deák F.u. 12.

C+4 programokat adok, cserélek magnón 8 Ft/db. Válaszborítékért lista.
Bujdosó Csaba, 2340 Kiskunlacháza, Tavasz u.7.

Olcsón kazettán C-64-re játékot adok el. Válaszborítékra listát küldök. A prog. ára: 10—20 Ft.
Rosenberg Ádám, 2100 Gödöllő, Szőlő u. 28.

Keresem a következő programokat: Total Recal, Indiana Jones, Predator, Double dragon. Superman, Pókember összes részét.
Rosenberg Ádám, 2100 Gödöllő, Szőlő u.28.

Katalógusáron (1500 Ft) eladó 6510-es mikroprocesszor C-64-hez.
Bánhalmi Péter, Miskolc, Szent István u. 19. Tel.: (46) 341-753.

Totókulcs-készítő programok C-64-re (8Kbyte-osak), melyek vezetéses játékot készítenek. Válaszborítékért tájékoztató.
Nagy Miklós, Újfehértó, Pf.:41.

C-64-re programokat eladók kazettán. 8-10 Ft/db. Kérésre listát küldök. Feladandó az újság címére: Commodore Újság, 1388 Budapest, Pf.. 86.

Jó állapotú C-16 + memóriabővítő, szakirodalom, játékok és új C-64 + 1541/II floppy, rengeteg szakirodalom és lemez játékokkal és egy hangdigitalizáló kedvező áron eladó.
Selmeci Marcell, 5000 Szolnok, Meder u. 24. III/11. Tel.: 32-891.

Commodore 128/64 új programlemezek eladók. Programmal 70, üresen 40 Ft/db 400 lemez a választék. Válaszbélyegért listát és tájékoztatót küldök.
Kopácsi Lajosné, 1031 Budapest, Vízimolnár u. 2. X/95.

Ez itt a FEFE'S Software! C-64-re floppyra. Írjál!
Krisztbaum Ferenc 1105 Bp. Ónodi köz 15.

Eladó áron alul egy Yamaha VSS200 szinti, egy Aiwa 300-as deck és egy C-64 + magnó + joystick + programok.
Regőczi István, 3300 Eger, Kiskanda u. 2. 3/28.

C-64-re cserélnék programokat kazettán és lemezen. Továbbá elcserélnék: 3 db gyári kazettát egy távirányítású joystickra.
Czifra Zoltán, 5231 Fegyvernek, Kiss J. u. 118/a.

C-64 programok lemezen olcsón eladók. Reset 300 Ft-ért, új DS/DD disck 35 Ft-ért eladó. Válaszborítékra listát küldök. Leveleket a szerkesztőségbe.

C-64 + 1541 floppy + joystick + programok + könyvek együtt olcsón eladó.
Danyi Tamás

C-64 programok lemezen nagy választékban eladók. Egy lemezoldal 20-30 Ft. Válaszborítékra listát küldök.
Nagy István, 1202 Budapest, Mártírok útja 151.

Eladó c-64 + floppy + játékprogramok (50 db).
Sipos István, 1046 Budapest, Székpatak u. 7. II/4. Tel.: 1220-660/715

Helló Mindenkinek! C-64-es magnósok most hozzátok szólok. Nem untátok már meg a régi programjaitokat? Ha meguntátok, akkor írjatok! Utántöltős programok 70 Ft/db áron, pl.:Shadow of the beast, Rolling Ronny, Supermacy, Shadow dancer, minden 5. program után egy Bonus program(ingyen). 15. program után extra Bonus pl.: Super Cars, Betrayal, Jai Alai. És ha még ez sem elég, minden 20. megrendelést ingyen teljesítek.
Gégény László, 4700 Mátészalka, Kézi L. u. 6. Tel.:(44) 12-811

Új C-64 + magnó + monitor + 20B joystick + játék programok eladók.
Ágyik Attila, 8666 Bedegkér, Fő út 74/a.

Infravörös, mikrókapcsolós, gyári új joystick commodore gépekhez 1800 Ft + postai utánvét.
Izbéki Tibor, 5452 Mesterszállás, Ady Endre út 9.

Elromlott C-64-es tápegységért postai utánvéttel, 10 napon belül 1500 Ft-ért megjavítom.
Izbéki Tibor, 5452 Mesterszállás, Ady Endre út 9.

AMIGA-kiegészítők olcsón eladók! Kérjen tájékoztatót!
Izbéki Tibor, 5452 Mesterszállás, Ady Endre út 9.

Eladó C-64 + VC1541 + 1joy. 18 000 Ft-ért. Érdeklődni: hétköznapokon 14-19-ig.
Salamon, 1039 Budapest, Sarkadi u. 3. IV/13.

Színvonalas programok C-64-hez DONALD DUCK stb.
Tóth Gábor, 5919 Pusztaföldvár, Rákóczi u. 43.

C-64 programok eladók 90 Ft/lemez. Válaszborítékra listát küldök.
Farkas István, 8900 Zalaegerszeg, Kelemen I. u. 16.

C-64-es programok eladók olcsón. Válaszborítékra listát küldök.
Kozma Gergely, 1144 Budapest, Szentmihályi u. 7.

C-64/II + OC-118 floppy + PROFEX magnó + 15 kazetta + 150 lemez/tartó + JOY eladó. KÜLÖN, KÜLÖN IS. UGYANITT SZÁMÍTÓGÉP, MONITOR, MAGNÓ DECK, ERŐSÍTŐ, VIDEÓ-hoz közvetlen csatlakozású kábelek készítése és javítása garanciával!
Patyik Béla, 6758 Röszke, Dózsa út 27.

Tagtársak figyelem! Itt a SULI-SOFT katalóguslemez, mely Commodore gépeken olvasható! Programleírások, ajándékprogramok! Két lemezoldalnyi információ, katalógusfüzettel együtt mindösszе 300 Ft + postaköltség. Megrendelhető: SULI-SOFT 1327 Újpest 3., PF.: 91.

C-64-re játék és felhasználói programok eladók. Válaszborítékért listát küldünk!
FRIEND TWO CREW,
1399 Budapest, Pf.: 701/55.

C-64 + 1541-II floppy + magnó + 3db joystic + cartridge + 10 kazetta + 42 lemez (összeesen 500 prg.) 32 000 Ft.
Tel.: 27/10-288.

Eladó 1531 C= magnó, C-64 tápegységek, originál 3M lemezek; 2 doboz TDK, 1 doboz MAXELL lemezek programokkal.
Tel.: 227-22 -60.

Commodore 1802 színes monitor, keveset használt, 15 000 Ft-ért eladó.
Tel.: 177-60-09.

Nyomtatószalagok újrafestése postai utánvéttel is. MPS 803: 155 Ft, MPS 801: 171 Ft, DFX 5000: 799 Ft.
Oláhné, 2092 Budakeszi, Erkel u. 31./c. Tel.: 1-121-127.

FORDÍTÓT KERESÜNK!
Az Országos Commodore Egyesület angol és német szakirodalomból fordító külső munkatársat keres.
Tel.: 1-76-22-57.

A NOVOTRADE SZERVÍZ Kft. az alább felsorolt szervízeiben mindenféle szervízszolgáltatás munkadíjából 10% kedvezményt ad az egyesületi tagoknak.

1053 Budapest, Magyar u. 12—14	Telefon: 117-3551
1083 Budapest, Szigony u. 9.	Telefon: 134-3153
1191 Budapest, Gábor Á. sétány 3.	Telefon: 127-4763
3525 Miskolc, Fazekas u. 1—3.	Telefon: 46-17-011
4034 Debrecen, Holló L. u. 14.	Telefon: 52-32-863
5600 Békéscsaba, Bartók B. u. 37.	Telefon: 66-27-195
6724 Szeged, Csongrádi sugárút 76.	Telefon: 62-13-377
7624 Pécs, Jurisics M. u. 17.	Telefon: 72-11-812
8000 Székesfehérvár, Széchenyi u. 15/a.	Telefon: 22-12-711
9700 Szombathely, Szalonok u. 31.	Telefon: 94-13-419

Felvevőhelyek:

9024 Győr, Babits M. 75.	
6000 Kecskemét, Széchenyi tér 1—3.	Telefon: 76—23—720

Igazolás: a javítandó berendezés leadásakor egyesületi igazolvánnyal.
A kedvezmény többször is igénybe vehető.

Novotrade SZERVÍZ Kft.

Az Országos Commodore Egyesület szolgáltatásai

Egyesületi tagoknak 20% kedvezmény:

VC—20 memóriabővítés 3—27 kByte-os:	kiépítéstől függő
C—16, C—116 memóriájának bővítése 64 kByte-ra:	3500 Ft
C—16 belső 16 kByte-os EPROM bővítés:	1450 Ft
C—16 belső 32 kByte-os EPROM bővítés:	2900 Ft
C—16 belső 8 kByte-os SOFT—ROM bővítés:	2800 Ft
C—16 belső 32 kByte-os SOFT—ROM bővítés:	4000 Ft
C—16 8 kByte-ról 32 kByte-ra átalakítás:	2000 Ft
C—16 és 1541 kompatibilis lemezegység párhuzamosítása:	3200 Ft
SOFTROM modul 32K, kikapcsoláskor sem felejt C-16, C-116, +4	5000 Ft
FÉK C—16, C—116, +4 potméteres sebességváltoztatás 0%-tól 100%-ig fokozatmentesen	2000 Ft
TTL IC-teszter (Cartridge+lemezen a program)	4300 Ft
+4, C—16, C—116 UNI—ROM modul különféle kiépítésekben:	
— 8 kByte SOFT—ROM	3400 Ft
— 16 kByte SOFT—ROM	4000 Ft
— 8 kByte SOFT—ROM 16 kByte EPROM	4400 Ft
— 16 kByte SOFT—ROM 16 kByte EPROM	5000 Ft
— 16 kByte EPROM	2200 Ft

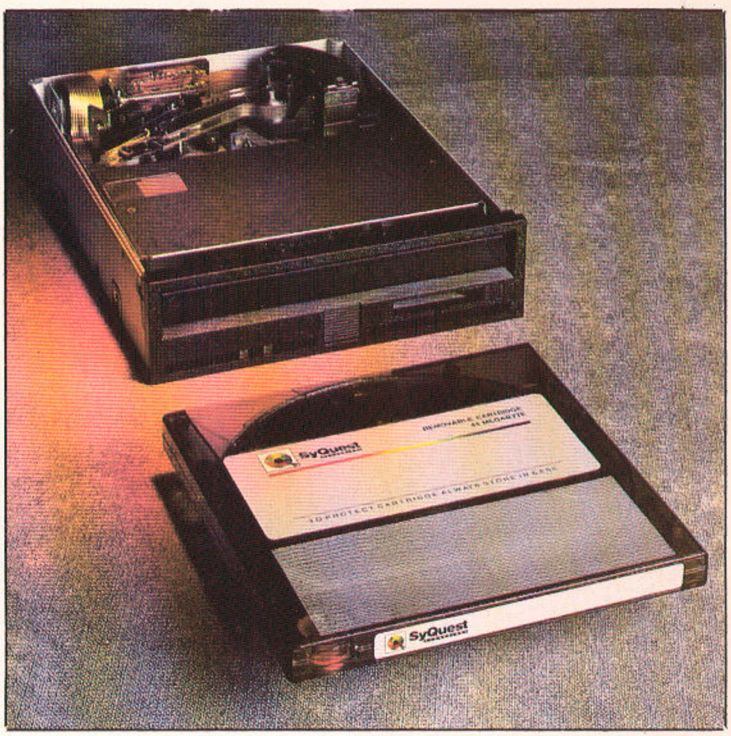
Kedvezmény nélkül:

C64-be átkapcsolható új operációs rendszer (Speed) + reset beépítése:	2000 Ft
1541 kompatibilis lemezegységbe Speeddos beépítése (átkapcsolhatóan) 40 TRACK (+ 85 blokk/lemezoldal), valamint párhuzamos 15 pólusú Canon csatlakozó beépítése:	2000 Ft
C64 USER-port 1541-es lemezegység összekötő párhuzamos kábel:	1300 Ft
1541 kompatibilis lemezegységbe elektronikus lemezlyukasztó beépítése:	800 Ft
PAGEFOX magyar ékezetes szövegszerkesztővel rendelkező cartridge: (Epson típusú nyomtató min. 640 képpontos szükséges a nyomtatáshoz)	6000 Ft
FASTLOAD (lemezes gyorstöltő, másoló, monitor)	1400 Ft
TTL IC-TESZTER cartridge + program	4300 Ft
288/256 Kbyte-os eprombank (vezérlő eprommal)	5340 Ft
Epromégető (2716-tól 27256-ig)	4300 Ft
8—16 Kbyte-os epromkártya (cartridge, eprom nélkül)	600 Ft
C64-hez tároló oszcilloszkóp (párhuzamos kábel nélkül)	7500 Ft
A háttértárakhoz epromok programozása (kész programok, vagy saját, hozott programok beégetésével) egységesen:	500 Ft
C64 bővítő-port elosztó (egyszerre 4 db cartridge lehet a gépben, melyeket gombnyomásra lehet kapcsolni)	7500 Ft
C64 USER — CENTRONICS nyomtatókábel (GEOS kábel)	1500 Ft
256K RAM-diszk	13000 Ft
64/256K RAM-diszk	9000 Ft
256K RAM-diszk (RAM-ok nélkül)	7500 Ft
64 Kbyte-os cartridge komplett programokkal, vagy igény szerint összeállítva	2500 Ft

A fenti bővítések megrendelhetők levélben, vagy az OCE irodájában személyesen.
Ha személyesen kívánja megrendelni, kérjük, előtte telefonáljon.

Áraink az alkatrészárak változásai befolyásolhatják.

Cserélhető lemezes winchester!



A SyQuest Technology hivatalos magyarországi disztribútora:

NOVOTRADE
SZERVIZ Kft.

Cím: 1053 Budapest, Henszlmann I. u. 9.
Telefon: 117-4144 Telefax: 117-9692

SQ 555 (meghajtó)	**39 900 Ft**
SQ 400 (44 MB/lemez)	**8 400 Ft**
SQ 5110 (meghajtó)	**61 400 Ft**
SQ 800 (88 MB/lemez)	**11 900 Ft**

Áraink végfelhasználói árak és ÁFA-t nem tartalmaznak.

Meghatalmazott dealereink:

Microteam Kft., 1145 Budapest, Róna u. 127.
Tel./fax: 184-1226

Professzionál Kft., 1033 Budapest, Kaszásdűlő u. 5.
Tel.: 167-0024 fax: 167-0289

Professzionál Kft., Miskolci Kirendeltség,
3525 Miskolc, Szabó L. u. 37.
Tel.: 46/56-079

Professzionál Kft., Békéscsabai Kirendeltség,
5600 Békéscsaba, Andrássy u. 75.
Tel.: 66/28-584

Korall Kft., 2800 Tatabánya, Március 15. út 3.
Tel.: 34/11-714

3S Computer Kisszövetkezet, 6723 Szeged, Kemes u. 6.
Tel.: 62/26-277, Fax: 62/26-347

Volánelektronika Vercomp Kft., 9024 Győr, Dr. Petz Lajos u. 7.
Tel./fax: 96/12-520

Novotrade Miskolc Kft., 3530 Miskolc, Vörösmarty u. 51.
Tel./fax: 46/49-489

Elektrosoft Kft., 5000 Szolnok, József A. u. 6—8.
Tel.: 56/42-880, Fax: 56/44-222

Számadó Kft., 6000 Kecskemét, Dózsa Gy. u. 29.
Tel.: 76/21-455, Fax: 76/21-462

BX-Next Kft., 3434 Mályi, Bercsényi út 50.
Tel.: 46/91-117

Onyx Szoftver Kft., 1118 Budapest, Mányoki u. 14/B.
Tel.:165-3325

Novotrade PC Kft., 1136 Budapest, Sallai u. 25.
Tel.: 149-0798 Fax: 131-0734

Almárium Kft., 1137 Budapest, Pozsonyi u. 21—23.
Tel.: 111-2830 Fax: 112-3647

Microchip Kft., 8000 Székesfehérvár, Élmunkás u. 47.
Tel.: 22/25-514

Navigátor Kft., 4400 Nyíregyháza, Kórház út 26/B.
Tel./fax: 42/41-972

Alfadat Kft., 2803 Tatabánya, Tóth-Bucsoki út 12.
Tel.: 34/10-234, 10-405, Fax: 34/10-729

Digitech Kft., 7101 Szekszárd, Rákóczi u. 6.
Tel.: 74/16-874

Interface Kft., 1116 Budapest, Hunyadi J. út 162.
Tel.: 166-5322/58, 55, Fax: 226-3793

Netrend Rt., 1089 Budapest, Elnök u. 1.
Tel.: 113-8217, Fax: 113-9537

Omnis Kft., 2840 Oroszlány, Münnich F. u. 23.
Tel.: 34/60-832

2R Periféria Kft., 1071 Budapest, Peterdy u. 30.
Tel.: 122-3034 Fax: 142-3308

ProComp Kft., 8900 Zalaegerszeg, Bíró M. út 8. Pf.: 275.
Tel.: 92/11-373

A magyar olimpiai csapat arany fokozatú támogatója

SZERENCSEJÁTÉK RT

Ára: 68 Ft

input: MAX output: maximum

A TUNGSRAM-MAX mágneslemez japán és amerikai alapanyagból, amerikai technológiával, high-tech berendezéseken készül. Minden egyes mágneslemez hibamentességét a teljes felület számítógépes mérőrendszerrel történő tesztelése garantálja.

TUNGSRAM-MAX mágneslemezek

Standard	Formázott	Színes lemezek műanyag dobozban
5,25" TM 2S2D 53 Ft	5,25" TMF 2S2D 61 Ft	5,25" TMP 2S2D 63 Ft
5,25" TM 2SHD 77 Ft	5,25" TMF 2SHD 87 Ft	5,25" TMP 2SHD 88 Ft
3,5" TM 2S2D 88 Ft	3,5" TMF 2S2D 102 Ft	3,5" TMP 2S2D 99 Ft
3,5" TM 2SHD 155 Ft	3,5" TMF 2SHD 173 Ft	3,5" TMP 2SHD 167 Ft
Árainkhoz ÁFA-t számítunk!		

- ☐ CSEREGARANCIA: 20 ÉV VAGY 20 MILLIÓ FORDULAT!
- ☐ Mágneslemezek: no name és bulk csomagolásban is!
- ☐ Tárolódobozok, tisztítókészletek 3,5" és 5,25" méretben.
- ☐ Szoftvermásolás profi gépeken, írásvédő kivágás nélkül is!
- ☐ Viszonteladóknak 20% engedmény!
- ☐ Szoftverkészítőknek, nagyfelhasználóknak, diákoknak rendkívüli kedvezmények!
- ☐ Színes és formattált mágneslemezek, tárolódobozok, festékszalagok árusítása, szoftvermásolás és csomagolás, címkézés a szoftverkészítő igénye szerint.
- ☐ Kérje részletes árlistánkat!
- ☐ AKCIÓS ÁRAK az IFABO-n az A pavilonban, 1992. április 27—30.
- ☐ Tungsram Magnetic Media
  Budapest IV., Váci út 77.
  Tel.: 160-2233 Fax: 160-0925



TUNGSRAM-MAX®

KÉPVISELETEINK:

Agromark Kft. — Hódmezővásárhely, Andrássy út 50. Tel.: 06-62-41695

Comtrade Kft. — Pécs, Majorossy u. 36. Tel.:06-72-26063

M és K Bt. — Kecskemét, Bajcsy-Zsilinszky u. 5. Tel.: 06-76-21878

PGM Computer Kft. — Szeged, Csongrádi sugárút 22. Tel.: 06-62-14380

Számker Kft. — Zalaegerszeg, Rákóczi út 4—8. Tel: 06-92-14500/144

Transzfer Kft. — Nyíregyháza, Hősök tere 7. Tel.: 06-42-10481

HIVATALOS DEALEREINK:

Novotrade 2C Kft. — Budapest XIII., Balzac u. 35. Tel.: 140-2954

Radiant Kft. — Budapest XIV., Francia út 11. Tel.: 252-1999/266

Westing Iroda — 1149 Budapest, Róna köz 12. Tel.: 163-7916

TUNGSRAM MAGNETIC MEDIA RT.
H-1340 Budapest, Váci út 77.
Tel.: 160-2233, Fax: 160-0925

TUNGSRAM MAGNETIC MEDIA

SZERENCSEJÁTÉK RT

A magyar olimpiai csapat arany fokozatú támogatója